David W. Taylor Naval Ship Research and Development Center
Bethesda, MD 20084-5000

CMLD-88/14 JUNE 1988

COMPUTATION, MATHEMATICS & LOGISTICS DEPT.
DEPARTMENTAL REPORT

COMPUTER CENTER REFERENCE MANUAL

DAVID V. SOMMER
SHARON E. GOOD

Approved for Public Release:
Distribution Unlimited

CMLD-88/14 COMPUTER CENTER REFERENCE MANUAL

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

1b. RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION AUTHORITY

2b. DECLASSIFICATION/DOWNGRADING SCHEDULE

3. DISTRIBUTION/AVAILABILITY OF REPORT
Approved for Public Release;
Distribution Unlimited

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
CMLD-88/14

5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION
DTRC

6b. OFFICE SYMBOL (If applicable)
1893

7a. NAME OF MONITORING ORGANIZATION

6c. ADDRESS (City,State,and ZIP Code)
Bethesda, MD 20084-5000

7b. ADDRESS (City, State, and ZIP Code)

8a. NAME OF FUNDING/SPONSORING ORGANIZATION

8b. OFFICE SYMBOL (If applicable)

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c. ADDRESS (City,State,and ZIP Code)

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Computer Center Reference Manual

12. PERSONAL AUTHOR(S)
David V. Sommer, Sharon E. Good

13a. TYPE OF REPORT
Final

13b. TIME COVERED
FROM 062088 TO indef

14. DATE OF REPORT (Year,Month,Day)
88/06/20

15. PAGE COUNT
425

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

CDC NOS	Control statements	DEC VAX/VMS
CDC NOS/BE	Conversion	Hardware
Computer	Cray COS	Interactive

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report provides an introduction to the operating systems of the Cray X-MP (COS), DEC VAXcluster (VMS), and CDC (NOS) for applications programmers. Some information has been distilled from many individual documents and augmented to reflect usage at DTRC. Control statement examples and descriptions of hardware and software are included.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED □ SAME AS RPT □ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL
David V. Sommer

22b. TELEPHONE(Include Area Code)
(301)267-3343

22c. OFFICE SYMBOL
1893.1

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

18 (continued)

Mass Storage System
Programming
Software Documentation
Supercomputer

David Taylor Research Center
Bethesda, Maryland 20084-5000

Computer Center
Reference Manual

by
David V. Sommer
Sharon E. Good

Software Branch
Code 1893

	Carderock	Annapolis
Phone	(202) 227-1907	(301) 267-3343
Autovon	287-1907	281-3343

For recorded message on computer status (202) 227-3043

Questions and requests for more detailed information should be directed to Code 1893, Bldg. 17, Rm. 226

Computation, Mathematics and Logistics Department
Departmental Report

June 1988 CMLD-88/14

.. Through Revision 0 (Oct 1988) ..

*** Revision Record ***

Revision	Description
0 (Oct 88)	Original printing.

This page is intentionally left blank.

Contents

Appendices

Abstract

The Computer Center in the Computer Facilities Division of the David Taylor Research Center has installed a Integrated Supercomputer Network. This manual provides an introduction to the new Network. Some information has been distilled from many individual documents and augmented to reflect usage at DTRC. Control statement examples and descriptions of hardware and software are included, as is information on moving files among the CDC CYBER 860A (with the Mass Storage System), the DEC VAXcluster, the DEC Remote minis, and the Cray X-MP, creating and executing batch jobs, and using the interactive systems.

Administrative Information

The work described in this report was performed in the Software Branch (1893) of the Computation, Mathematics and Logistics Department, David Taylor Research Center, under the sponsorship of the DTRC Computer Center (189).

***** Introduction *****

The DTRC Integrated Supercomputer Network consists of a Cray X-MP/24 with 5 front-end computers: the DEC VAXcluster (four processors: two VAX 8550s and two VAX 11/780s), and a CDC CYBER 180/860A. The Cray and VAXcluster can store and retrieve files on the Mass Storage System (MSS), which is part of the CDC CYBER 860A. There may be several mini-sites, each with local processing capability as well as access to the Central Site computers. One mini-site is at Annapolis.

The following operating systems are in use:

Cray X-MP	COS version 1.16
DEC VAXcluster	VMS version 4.6
DEC Remote Mini (Annapolis)	VMS version 5.0
CDC CYBER 860A	NOS version 2.5.3

The front-end computers support both batch processing of jobs submitted at central site, through remote batch terminals or from interactive terminals; and demand processing, which supports a variety of interactive terminals. In addition, batch jobs can be sent to the Cray for processing with the output returned for examination or printing.

This reference manual is designed to provide the new user with enough information to use the Network to run simple batch jobs and to create and run programs and batch jobs interactively. Most of the frequently used control statements are described in detail in Appendices B and D. Magnetic tapes are discussed briefly. No attempt is made to describe all features of the operating systems or even all parameters of the control statements presented. More information can be found in the publications listed in Appendix F.

Before using the system, job order number(s) to be charged must be registered with Code 189.3. Outside users must transfer funds to DTRC before receiving a job order number. Each individual user should have 4-character User Initials assigned (also by Code 189.3).

*** Hardware Configuration ***

** Cray X-MP / 24 **

Cray station ID: C1

```
  2 X-MP central processing units (117 MFLOPS each)
 4M 64-bit words of central memory
  4 model DD-49 disk storage units (4.8 Gbytes)
```

** CDC CYBER 180 model 860A **

Network ID: MFN
Cray station ID: N1

```
  1 CYBER 860A central processing unit (6.3 mips)
 2M 60-bit word memory
 25 peripheral processors
  3 model 895   disk drives
  4 model 679-5 nine-track tape drives (1600/6250 cpi)
  2 model 679-3 nine-track tape drives ( 800/1600 cpi)
  2 model 677-3 seven-track tape drives
  1 model 405   card reader
  1 model 415   card punch
  2 model 585   line printers (1200 lpm, upper/lower case)
  1 model 7990  Mass Storage System (210 Gbytes)
    3 model M861 storage modules
  1 CDCNET communications system
     16 dial-up lines for ASCII/BCD 4800-baud terminals
                                             (202) 227-4740
     56 dial-up lines for 1200-/300-baud interactive terminals
                                             (202) 227-4800 (32)
                                             (202) 227-4850 (16)
                         Annapolis - x4741 or x4761 then 56  (8)
```

** DEC VAXcluster **

VAXcluster nodes: DT1, DT2, DT3, DT4
Cray station IDs: V3
V1, V2, V4 (future)

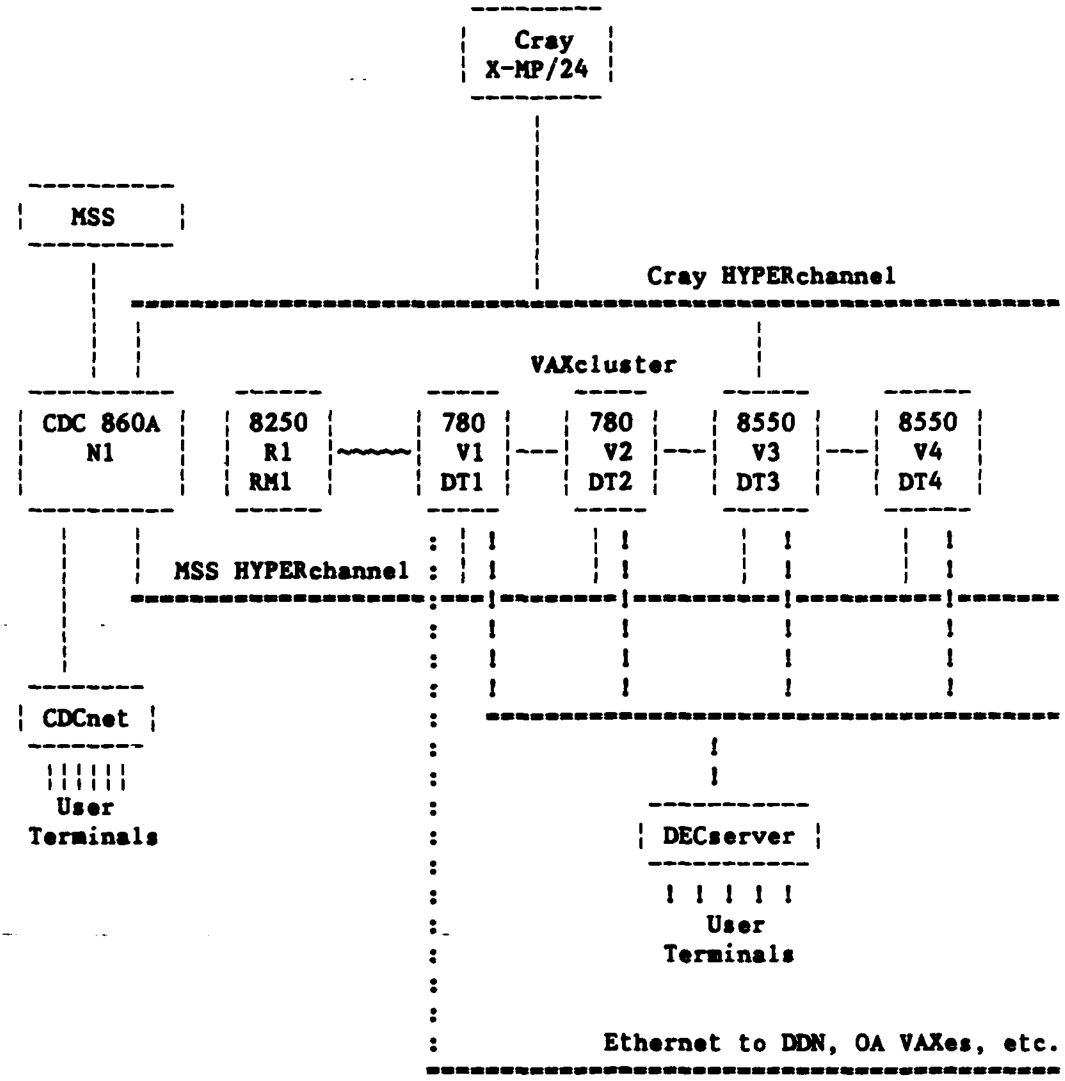
2 VAX 11/780 processors (1 mips each; DT1, DT2) -- each with 16 Mbyte central memory
2 VAX 8550 processors (6 mips each; DT3, DT4) -- each with 48 Mbyte central memory
2 model SA482 disk storage array (5.0 Gbytes)
9 model RA81 disk drives (4.1 Gbytes)
1 model TA79 nine-track tape drives (1600/6250 cpi)
3 model TU79 nine-track tape drives (1600/6250 cpi)
2 model TA78 nine-track tape drives (1600/6250 cpi)
3 model LP27 impact printers (800 lpm, upper/lower case)
1 model LP11 impact printer (300 lpm, upper case)
1 DECserver 500 network terminal switch
56 dial-up lines for 4800-/1200-/300-baud interactive terminals
(202) 227-5600 (48)
Annapolis - x4741 or x4761 then 57 (8)

** DEC Remote Mini **

DECnet node: RM1
Cray Station ID: R1 (future)

1 VAX 8250 processor (1.2 mips)
16 Mbyte central memory
1 model RA81 disk drive (450 Mbytes)
2 model TU81 nine-track tape drives (1600/6250 cpi)
1 model LP27 impact printer (800 lpm, upper/lower case)
8 dial-up lines for 4800-/1200-/300-baud interactive terminals
Annapolis - x4741 or x4761 then 58

*** The Integrated Supercomputer Network ***



***** User Interface With the Computer Center *****

*** General Information ***

The ADP Control Centers are located at Central Site at Carderock and Annapolis. You may submit decks and pick up output as well as obtain information on the progress of your jobs from an ADP Control Center.

Computer Center Notes is a publication sent to all registered users whenever there is information to be disseminated. The date of the latest news update is printed at the start of each batch job (log) or interactive session. The NEWS command or procedure is used to see the current news file.

*** Registering ***

To register to use Computer Center computers, call our Business Office, Code 189.3, at (202) 227-1361/1910. Be prepared to supply

- your name
- your DTRC code or non-DTRC company name and address
- the job order number(s) to be charged for computer work
- the computers on which you which to be registered
  - Cray X-MP
  - DEC VAXcluster
  - VAX at Annapolis (RM1)
  - CDC CYBER 860

Registration for the DEC VAXcluster or Cray X-MP includes registration for the Mass Storage System (CDC CYBER 860).

You will be given

- User Initials (if you are a new user)
- the initial passwords (which MUST be changed during your first session) for each computer system for which you registered

*** Passwords, Passwords, Everywhere ***

Each computer system has its own password to gain access to it (the CDC CYBER 860A has two: one for interactive, one for batch). You MUST change these during your first session on each or you will be denied future access. For security, you are strongly urged to change your access passwords as soon as you can log into each computer. Passwords for all our computers expire in 90 days.

To change your access passwords, use

- Cray X-MP (password is 4-15 characters)

  - on the Cray

    ```
    ACCOUNT,AC=joborderno,UPW=current_pw,NUPW=new_pw.
    ```

  - from the VAXcluster

    ```
    @VSYS:NEWCRAYPW  current_pw  new_pw  new_pw  ac  wait
    ```

  See also page 2-1-7.

- DEC VAXcluster (password is 6-12 characters)
  DEC VAX (RM1)

    ```
    SET PASSWORD      <-- you will be prompted for your current
                          and new passwords
    ```

- CDC CYBER 860A / MSS (password is 4-7 characters)

  - on the 860 (interactive and batch -- to change both, enter interactively and in a batch job)

    ```
    PASSWOR,current_pw,new_pw.    <-- batch or interactive
      -or-
    PASSWOR.          <-- you will be prompted for your current
                          and new passwords (interactive only)
    ```

  - from the VAXcluster (both passwords)

    ```
    HFT PASSWORD      <-- you will be prompted for your current
                          and new passwords
    ```

*** Trouble Forms ***

A Trouble Form is used:

1) for refund requests
2) when problems are encountered
3) for suggestions, gripes and complaints.

The Trouble Form should include a succinct description of the problem and include as much documentation (dayfile or log, listings, dumps) as possible. It should be submitted to Code 1893.1 for processing.

Trouble Forms may be entered directly into the computer from any of the front-ends (VAXcluster, CYBER 860) using the GRIPE command. If supporting documentation is needed, please send it to Code 1893.1 (User Services).

*** Refunds ***

Requests for refunds on lost time must be accompanied by output of the run and a Trouble Form, and must be reported within five working days. Decisions on refunds will be made by Code 189.

*** ADP Control Center ***

The ADP Control Center has the following capabilities:

1) Clean, test and degauss magnetic tapes.

2) Process Calcomp plots.

The following EAM facilities are available off-line:

1) A small card interpreter is available at Central Site

2) Shredder (available at Central Site)

See Appendix G for Computer Center telephone numbers.

***** Software Available *****

The following table lists the major software products and the computers where they are available. Type "HELP @CCF Software" on the VAXcluster for the latest version of this table.

	DT1	DT2	DT3	DT4	Cray	CDC
	11/780	11/780	8550	8550	X-MP	860A
ABAQUS	x	x	x	x	x	-
ACSL	-	-	x	x	-	-
ALGOL	-	-	-	-	-	x
APL	-	-	x	x	-	x
APT	-	-	-	-	-	x
Basic	x	x	x	x	-	x
C (CC)	-	-	x	x	-	-
Calcomp	x	x	x	x	-	x
CDD	x	x	x	x	-	-
Cobol (1)	x	x	x	x	-	x
Datatrieve	x	x	x	x	-	-
DBMS	-	-	x	x	-	-
DECalc	x	x	-	-	-	-
DISSPLA	x	x	x	x	x	(2)
EISPACK	x	x	x	x	x	(2)
FMS	x	x	x	x	-	-
Fortran (1)	x	x	x	x	x	x
FTP	x	-	-	-	-	-
GPSS	-	-	x	x	-	x
Hasp	-	-	-	-	-	x
HOTSPOT	-	-	-	-	-	x
IMSL	x	x	x	x	(2)	x
INGRES	-	-	x	x	-	-
Kermit (1)	x	x	x	x	-	x
LINPACK	x	x	x	x	x	x
Macsyma	x	x	-	-	-	-
Nastran	-	-	-	-	x	-

(1) - also on RM1 (VAX 8250 in Annapolis)
(2) - coming

	DT1	DT2	DT3	DT4	Cray	CDC
	11/780	11/780	8550	8550	X-MP	860A
Pascal	x	x	x	x	x	x
Patran	x	x	x	x	-	-
PCA	-	-	x	x	-	-
Pert Time	-	-	-	-	-	x
PL/I	x	x	x	x	-	-
PLOT10	x	x	x	x	-	-
Proj Mgt (PM)	-	-	x	x	-	-
Rim	x	x	-	-	-	-
Simscript	-	-	-	-	-	x
SMP	x	x	x	x	-	-
SPY	-	-	-	-	x	-
TELNET	x	-	-	-	-	-
WIN/TCP	x	-	-	-	-	-
XMODEM	-	-	-	-	-	x

***** The Cray X-MP *****

The Cray X-MP/24 at DTRC is a powerful, general purpose computer having two central processing units (CPUs) which share files and are linked together. These CPUs share 4 million 64-bit words of memory. Each CPU achieves its extremely high processing rate (up to 117 MFLOPS (million floating point operations per second)) using its scalar and vector capabilities.

*** COS Version 1.16 ***

The operating system for the Cray X-MP at DTRC is the Cray Operating System (COS), version 1.16BF2, which supports both batch and interactive processing.

*** Accessing the Cray X-MP ***

Batch jobs are normally submitted from one of the front-ends using: CRAY SUBMIT on the VAXcluster, or CSUBMIT on the CDC CYBER 860A. They may also be submitted from a running batch or interactive job using the Cray SUBMIT command.

Interactive access is also from one of the front-ends using: CRAY INTERACTIVE on the VAXcluster, or ICF (Interactive Cray Facility) on the CDC CYBER 860A.

Both modes of access are described later in this chapter.

*** Cray Datasets ***

On the Cray, information is organized by COS into datasets, which may be on disk, memory-resident, or interactive. A dataset contains one or more files and may be temporary (available only to the job that created it) or permanent.

Each dataset has a disposition code to tell COS what to do with it when it is released. The 2-character alphanumeric disposition codes include SC (scratch - default), PR (print), IN (input), and ST (stage to the front end).

Jobs access local datasets, which may be temporary or permanent. Permanent datasets are made local by the ACCESS statement. Front end files are made local by the FETCH statement.

*** Changing your Cray password ***

Your Cray access password may be changed from a batch job or interactively on the Cray, or from a procedure on the VAXcluster which creates and submits a Cray batch job for you.

Batch:

```
    $ CRAY SUBMIT mynewpw.job

    where your file MYNEWPW.JOB contains:

    JOB,JN=ssss.
    ACCOUNT,AC=ac,US=us,UPW=current_pw,NUPW=new_pw.
```

Interactive:

```
    $ CRAY  INTER  /JN=jobname  /US=username
    !ACCOUNT,AC=ac,US=us,UPW=current_pw,NUPW=new_pw.
    !^Z                     <-- ctrl-Z
    CRAY> QUIT
    CRAY> EXIT
    $                       <-- you are back in DCL

        -or-

    $ CRAY INTER
    CRAY> Jobname: ssss
    CRAY> Username: ABCD
    ! as above
```

Via DCL procedure NEWCRAYPW:

```
    $ @VSYS:NEWCRAYPW  current_pw  new_pw  new_pw  [ ac ]  [ wait ]

    where new_pw is entered twice for verification

           ac     is your Cray account number (may be omitted if it
                     is the same as your current VMS login)

           wait   is WAIT           - wait for the job to complete and
                                      display the .CPR file
                     anything else  - to let the job run on its own
                                      (you will have file NUCRPW.CPR
                                      when it completes)

    This procedure creates and deletes temporary file N$U$P$W.JOB.
```

*** Batch Jobs ***

Cray batch jobs are very similar to CDC batch jobs, but with different terminology. A batch job consists of one or more files. The first file is the JCL control statement file. It is followed by source or data files as needed by the JCL file. A typical job consisting of one source and one data file (*) looks like this:

```
JOB,JN=jobname,....
ACCOUNT,AC=job_order_number,US=username,UPW=password.

   <JCL statements>

/EOF                                <-- end-of-file

   <source file>

/EOF                                <-- end-of-file

   <data file>

<eod>                               <-- end-of-data
```

A Cray batch job has at least four datasets:

@CS - the control statement file
(part of $IN, but not accessible to the user)

$IN - the job input dataset. Accessible by its local name, $IN, or as Fortran unit 5.

$OUT - the job output dataset. Accessible by its local name, $OUT, or as Fortran unit 6.

$LOG - a history of the job. Not accessible to the user. $LOG is appended to $OUT when the batch job terminates.

(*) - When executing several programs or one program several times, the /EOF is required only when a program reads until end-of-file. If a program reads a specific number of data records, or has its own pseudo-end-of-file, the /EOF must NOT be present.

** Batch Job Classes **

Batch jobs fall into four service classes: EXPRESS, NORMAL, DEFER, and SECURE. Charges for EXPRESS and SECURE are 50% and 25% higher, respectively, than for NORMAL; DEFER class charges are 30% less than NORMAL. To specify the EXPRESS or DEFER job class, use US=EXPRESS or US=DEFER on the JOB statement. NORMAL is the default (provided the job meets the time and memory requirements). SECURE jobs may be submitted only during secure time (see below).

There are time and memory restrictions on the EXPRESS and NORMAL service classes. Jobs which request more are downgraded one or more classes. DEFER and SECURE jobs have no such restrictions.

The following chart shows for each job class: its priority, the maximum number of such jobs to be allowed to execute at the same time, and the maximum time (in decimal seconds) and memory (maximum field length in decimal words) requirements.

			requirements	
class	priority	maximum # jobs	time	memory
EXPRESS	10	15	T < 1800	MFL < 512K
			T < 600	MFL < 1536K
			T < 60	up to max MFL (3532800)
NORMAL	8	10	T < 10800	MFL < 512K
			T < 3600	MFL < 1536K
			T < 600	up to max MFL (3532800)
DEFER	6	3	T < 200000	up to max MFL (3532800)
SECURE	-	-	T < 200000	up to max MFL (3532800)
Interactive	14	14		

** SECURE Batch Job Class ***

Classified processing may be done on the Cray X-MP by making prior arrangement with Operations. When run in "secure mode", node DT3 is removed from the VAXcluster. Access to the Cray is available only from a terminal in the computer room. Batch jobs submitted to the Cray from this terminal must have "US=SECURE" in the job statement; jobs with US=EXPRESS, US=NORMAL, US=DEFER, or with US= omitted, will be rejected. (Jobs with US=SECURE may not be submitted during unclassified time.)

** From the VAXcluster **

To use the Cray from the VAXcluster, log in to a node which can access the Cray, prepare your Cray batch job using any editor, and submit the job file(s) to the Cray using the CRAY SUBMIT command:

```
    $ CRAY SUBMIT filename
```

or

```
    $ CRAY SUBMIT file1,file2,...
```

```
where filename is a VAXcluster file containing the Cray job
                  (default file extension: .JOB)

      filei    is a VAXcluster file containing part of the Cray job
                  file1 - the job control statements
                  file2 - the next file in the job
                          (perhaps a Fortran source program)
                  file3 - the next file in the job
                          (perhaps the data for running the program)
```

The output will be returned to your file jobname.CPR, where jobname is taken from the job statement of the Cray job (JN parameter). See Appendix C: CRAY SUBMIT to print the output directly on a VAXcluster printer.

Files sent to the Cray must not have embedded tabs. See Appendix C: DETAB.

* VAXcluster-to-Cray Examples *

1) $ CRAY SUBMIT JOB1

where JOB1.JOB contains:

```
JOB,JN=MYJOB.                                      (1)
ACCOUNT,US=username,UPW=password,AC=account.       (1)
CFT.                                               (1)
SEGLDR,GO.                                         (1)
/EOF
       PROGRAM ADD                                 (2)
       DO 10 I=1,5                                 (2)
          READ (5, *) N1, N2, N3                   (2)
          N = N1 + N2 + N3                         (2)
          WRITE (6, *) N1, N2, N3, N             (2)
    10 CONTINUE                                    (2)
       END                                         (2)
/EOF
1  2  3                                            (3)
4  5  6                                            (3)
7  8  9                                            (3)
10 11 12                                           (3)
13 14 15                                           (3)
/EOF
```

will submit the job to the Cray with the output returned in file MYJOB.CPR.

2) $ CRAY SUBMIT RUN2.JOB,RUN2.FOR,RUN2.DAT

```
where RUN2.JOB contains the job control statements ((1) above)
      RUN2.FOR contains the Fortran source program ((2) above)
      RUN2.DAT contains the data                   ((3) above)
```

will submit the combined files to the Cray with the output returned in file MYJOB.CPR. Note that the /EOF records are not required in this format.

3) $ CRAY SUBMIT RUN3

where RUN3.JOB contains:

```
JOB,JN=MYJOB.
ACCOUNT,US=username,UPW=password,AC=account.
FETCH,DN=PROG3,TEXT='PROG3.FOR'.
FETCH,DN=DATA3,TEXT='PROG3.DAT'.
CFT,I=PROG3.
SEGLDR,GO.
```

PROG3.FOR on the VAXcluster contains the program (2) above, with "OPEN (5, FILE='DATA3')" before the "DO 10 ..."

PROG3.DAT contains the data (3) above.

** From the CDC CYBER 860A **

To use the Cray from the CYBER 860A, log in, prepare your Cray batch job using any editor, and submit the job file to the Cray using the CSUBMIT command (see Appendix D for additional parameters):

```
/CSUBMIT,lfn.            <-- print at Central Site
/CSUBMIT,lfn,RB=ANAP.    <-- print at remote batch terminal ANAP
/CSUBMIT,lfn,RB=un.      <-- put into output queue for user un
/CSUBMIT,lfn,TO.         <-- put into your output queue
```

In the last two formats, use QGET to get the file from the queue. To send the output elsewhere, use the Cray DISPOSE command (see Appendix B).

* CYBER 860A-to-Cray Examples *

1) /CSUBMIT,RUN1.

where local file RUN1 contains:

```
JOB,JN=myjob.
ACCOUNT,US=username,UPW=password,AC=account.
FETCH,DN=prog3,SDN=myprog,TEXT='GET,myprog.CTASK.'.
                                                ^-- indirect file
FETCH,DN=mydata,TEXT='ATTACH,mydata.CTASK.'.    <-- direct file
CFT,I=prog3.
SEGLDR,GO.

   -or-

JOB,JN=myjob.
ACCOUNT,US=username,UPW=password,AC=account.
FETCH,DN=prog3,SDN=myprog,TEXT='GET,myprog.CTASK.'.
                                                ^-- indirect file
ACCESS,DN=PROCLIB,OWN=PUBLIC.
LIBRARY,DN=PROCLIB:*.
MSACCES,US=user,MPW=mspass.
MSFETCH,DN=mydata.                              <-- direct file
CFT,I=prog3.
SEGLDR,GO.
```

MYPROG and MYDATA on the CDC CYBER 860A contains the program and data (see page 2-1-6, example 3).

** From a Running Cray Job **

A batch job to be submitted from a running Cray job may reside either on the Cray or on one of the front-ends. From within the Cray job, ACCESS or FETCH the file to make it a local file, then SUBMIT it to the COS input queue. (See Appendix B for additional parameters for these Cray commands.)

* Examples *

1) The job is in a permanent dataset on the Cray:

```
JOB,....
ACCOUNT,....
...
ACCESS,DN=myjob,PDN=mypermjob.
SUBMIT,DN=myjob.
...
```

2) The job is in a file on the VAXcluster:

```
JOB,....
ACCOUNT,....
...
FETCH,DN=myjob,TEXT='myjob.job'.    <-- submitted from VAXcluster
   -or-
FETCH,DN=myjob,MF=V3,TEXT='DT3"user pw"::U0n:[user]myjob.job'.
                                    ^-- submitted from CYBER 860
                                        or VAXcluster
SUBMIT,DN=myjob.
...
```

3) The job is in a file on the Mass Store (CDC CYBER 860A):

```
JOB,....
ACCOUNT,....
...
ACCESS,DN=PROCLIB,OWN=PUBLIC.
LIBRARY,DN=PROCLIB:*.
MSACCES,US=user,MPW=mspass.
MSFETCH,DN=myjob.
SUBMIT,DN=myjob.
...
```

*** Interactive Jobs ***

Cray X-MP interactive access is via the Cray Station code on one of the front ends.

** From the VAXcluster **

The Cray X-MP is accessed via the VMS Cray Station, which may be entered by the CRAY command. The INTERACTIVE Station command allows interactive use of the Cray. You enter the Cray Station, request interactive service, do your thing, leave Cray interactive, terminate the Cray session (from the Cray Station), and leave the Cray Station. You will then be at the VMS prompt.

The CRAY command puts you into Cray context (indicated by the CRAY> prompt).

Type CRAY INTERACTIVE, or CRAY and then INTERACTIVE at a CRAY> prompt. You will then be requested to supply:

```
    CRAY JOBNAME:       <-- enter anything you wish as the jobname for
                            this session
    CRAY USERNAME:      <-- enter your User Initials
```

You will then be connected to the Cray itself, which has an exclamation prompt (!). Your first command must be your ACCOUNT statement. Any other commands will be ignored until a valid ACCOUNT statement is read.

```
    !ACCOUNT,AC=1222233344,UPW=pw,US=userinit.
```

When you receive another ! prompt, your logon was successful. You may now use any of the commands in Appendix B. Every command MUST end with a terminator (.); if you forget, use the up-arrow to bring the command back and add the terminator.

To leave Cray interactive temporarily, enter an end-of-file (^Z). This brings you back to the Cray Station where you can do any Station command.

To terminate the Cray interactive session, enter the Station command QUIT. You are still in Cray context and can enter any Station command. It is recommended that you use the STATUS command to be sure your interactive session terminated. If it didn't, enter "KILL jsq".

To leave the Station, enter EXIT (or ^Z). This will bring you out of Cray context and back to the VMS prompt.

* VMS Cray Station Commands *

See Appendix C for the syntax of these commands.

$ Create a temporary VMS subprocess, allowing you to enter DCL commands. To return to Cray context, type LOGOUT.

+ Display the next page of information in Cray context.

- Display the previous page of information in Cray context.

@ Execute an indirect station command file (containing station commands) in Cray context. (Synonym for PLAY.)

^Z CTRL-Z exit the current processing mode. In response to the Cray context prompt (CRAY>), it returns you to DCL; during a Cray interactive session, it returns you to command mode. While you are being prompted for command parameters, CTRL-Z cancels the command. You can also terminate the execution of an indirect station command file with CTRL-Z.

ABORT Interrupt the current interactive Cray job step and return to the "!" prompt after first displaying any COS output queued for the terminal.

ATTACH Redirect COS interactive terminal output to an alternate device.

ATTENTION Interrupt the current interactive Cray job step and enters reprieve processing. If no reprieve processing, ATTENTION is the same as ABORT.

BYE Terminate an interactive session. Depending on the command qualifiers, the COS interactive job may also be terminated.

CINT Enter a subset of Cray context that incorporates only the INTERACTIVE command and its associated subcommands. No other Cray context commands are available during a CINT session. CINT is designed to give better interactive performance, since it invokes only a subset of the Cray context image. CINT is available at DCL level. Use INTERACTIVE (in Cray context) for the full set of Cray context commands.

CLEAR Terminate any display command and clear the display portion of the screen.

COLLECT Store COS interactive output in a VMS file.

COMMENT Insert comments into an indirect station command file stream.

CRAY Enter the Cray context utility or execute a single station command when that command is supplied as a parameter. If a command parameter is not included after the CRAY command, you remain in Cray context until you enter the EXIT command.

DATASET Report the existence of a COS permanent dataset.

DELAY Suspend execution of an indirect station command file for a specified period of time.

DISCARD Discard all output from a COS interactive session until the next COS prompt is issued.

DROP Terminate a COS job and returns the associated output dataset. COS job execution enters reprieve processing after the next COS EXIT control statement.

EOF Send an end-of-file record to a connected COS interactive job. This command is normally required to terminate COS file input from the terminal.

EXIT Return you from Cray context command mode to DCL command state. If you issued a RECORD command during the session and the specified file is still open, the file is closed.

HELP Display information from the station help files or an index of all commands.

INTERACTIVE Initiate or restart an interactive session.

ISTATUS Return the status of your COS interactive job, including the CPU time used and the last COS logfile message.

JOB Display the status of a specific COS job.

JSTAT Display the status of a specific job and its related tasks.

KILL Terminate a job immediately.

LOGFILE Provide access to the station logfile messages.

LOOP Restart execution of an indirect station command file at the beginning of the file. End looping with Z.

MESSAGE Send a message to the COS job and station logfiles.

PAUSE Suspend the execution of an indirect station command file. Control is passed to the terminal, where you can terminate the command file by entering a command or resume it by entering a null line (<RET>).

PLAY Execute an indirect station command file. (Same as @.)

QUIT Terminate a Cray interactive session and the corresponding COS interactive job. (Equivalent to BYE/ABORT.)

RECORD Start or stop the recording of terminal input to the specified file while in Cray context for later use with the PLAY or @ commands. Exiting Cray context automatically issues a RECORD/OFF.

RELEASE Release a dataset that is held by COS.

REMOVE Delete entries in the dataset staging queue.

RERUN Immediately end the processing of a COS job and put job back into the input queue, unless the job has terminates or cannot be rerun.

SAVE Stage a VMS file to COS permanent file.

SET Define terminal working environment for the current session.

SHOW Display information about the status of the station staging queue.

SNAP Copy the current contents of the display region into the specified VMS file. If the command is issued from a terminal in line-by-line mode, the last display requested is recorded in the file.

STATCLASS Display the current COS job class structure.

STATUS Display the COS system status.

STORAGE Initiate a COS mass storage status display providing the following information: device class or status; device name as it is known to COS; percentage of free space and permanent space on each device; number of recovered and unrecovered errors on each device; location of last error.

SUBMIT Stage the specified VMS file to COS to be put on the job input queue. The file must contain COS JCL (see HELP @COS). The first record must be the JOB control statement. By default, the output from the COS job (known as a logfile) is sent to the directory from which the job was submitted.

SWITCH Set or clear COS job sense switches.

* Examples *

```
1) $ cray
   CRAY> interactive
   CRAY JOBNAME: abcd001                <-- any jobname
   CRAY USERNAME:ABCD                   <-- user ABCD
   !ACCOUNT,AC=1222233344,UPW=mypw.     <-- US=abcd not needed since
                                            upper case was used in
                                            entry above
   !<your Cray commands>
   !^Z                                  <-- ctrl-Z to leave Cray
   CRAY> <Station commands>
   CRAY> quit                           <-- terminate Cray session
   CRAY> status                         <-- be sure your session
                                            terminated
 ( CRAY> kill <jsq>                     <-- if STATUS show your session
                                            is still active )
   CRAY> <Station commands>
   CRAY> exit                           <-- leave Cray Station


2) $ cray interactive
   CRAY JOBNAME: struct                         <-- any jobname
   CRAY USERNAME:efgh                           <-- user EFGH
   !ACCOUNT,AC=1222233344,UPW=mypw,US=efgh.     <-- US=efgh needed
                                                    since lower case
                                                    was used in entry
                                                    above
   <same as example 1>


3) $ cray interactive /jn=struct /us=efgh       <-- any jobname;
                                                    user EFGH
                                                    (since this is a
                                                    VMS DCL command,
                                                    it is converted to
                                                    upper case)
   !ACCOUNT,AC=1222233344,UPW=mypw.             <-- US=efgh not needed
                                                    since the VMS
                                                    control statement
                                                    is converted to
                                                    upper case

   <same as example 1>
```

** From the CDC CYBER 860A **

The Cray X-MP is accessed via the NOS Interactive Cray Facility (ICF), which may be entered by the APPSW,ICF command from IAF. You enter ICF, log onto the Cray, do your thing (Cray or ICF commands), leave the Cray and ICF. You will then be at the NOS prompt.

Alternatively, you can specify ICF as the application when you log into NOS.

ICF commands have a prefix (normally a slash "/") and can be intermixed with Cray commands. To terminate the Cray session (and ICF), enter /BYE or /LOGOFF.

* NOS ICF User Commands *

/ABORT Send abort interrupt to the interactive Cray job (also user-break-2 key (normally %2).

/ATTENTION Send attention interrupt to the interactive Cray job (also user-break-1 key (normally %1).

/BYE Terminate this Cray interactive session. (Same as /LOGOFF)

/CONNECT Create a logical connection between this terminal and some other (slave) terminal.

/DISCARD Discard output being sent from the Cray to this terminal.

/ENDCONNECT Terminate a CONNECT.

/ENDPLAY Terminate reading of a PLAY file.

/EOF Send an end-of-file to the Cray.

/HELP Display help information.

/ICFSTATUS Display general information about the current status of ICF.

/LOGOFF Terminate this Cray interactive session. (Same as /BYE)

/LOGON Initiate or reconnect to an existing Cray job.

/PERIOD Set/reset automatic generation of a terminator on COS commands.

/PLAY Read data and commands from a NOS file in the user's catalog.

/PREFIX Change the ICF command prefix letter.

/QUIT Immediately terminate this Cray interactive session.

/RESUME Resume the transmission of data to and from the Cray (negate the effect of SUSPEND).

/SUSPEND Suspend transmission of data to and from the Cray.

/STATUS Display Cray status.

/* An ICF comment line.

* Examples *

```
1) /appsw,icf                          <-- / is the NOS prompt
   <a greeting>
   /logon mf=mcr                       <-- / is required;
                                           log onto DTRC Cray
   <a greeting>
   !account,ac=1222233344,upw=mypw.    <-- US=abcd not needed
   !<your Cray or ICF commands>
   !/bye                               <-- to leave Cray and ICF


2) FAMILY: ,abcd,pw,icf                <-- log into ICF directly
   <a greeting>
   /logon mf=mcr                       <-- / is required;
                                           log onto DTRC Cray
   <a greeting>
   !account,ac=1222233344,upw=mypw.    <-- US=abcd not needed
   !<your Cray or ICF commands>
   !/bye                               <-- to leave Cray and ICF
   T1210   - APPLICATION: iaf          <-- switch to another
                                           application such as IAF
```

***** Cray JCL Commands *****

The Cray Job Control Language (JCL) statements are grouped by function in this section. See Appendix B for a description of the syntax for each command. (DTRC) indicates a command or program added at DTRC. Some of the logic structure commands use JCL expressions, which are described later in this section.

*** Job Definition and Control ***

* Entire line is a comment.

ACCOUNT Validate a user's Job Order Number, user name and password.

ALTACN Validate an alternate account number for permanent datasets.

CALL Read control statements from another file.

CHARGES Report on job resources.

ECHO Control logfile messages.

EXIT On job abort, processing continues with the statement following the EXIT; if no abort, terminate job processing.

IOAREA Control access to a job's I/O area (containing the DSP and I/O buffers).

JOB First statement of a job -- gives job parameters.

JOBCOST (DTRC) Write a summary of job cost and system usage to $LOG.

LIBRARY Specify search order for procedures during processing.

MEMORY Request new field length.

MODE Set/clear mode flags.

NORERUN Control a job's rerunability.

OPTION Specify user-defined options.

RERUN Control a job's rerunability.

RETURN Return from an alternate control statement file.

ROLLJOB Protect a job by writing it to disk.

SET Change value of a JCL symbolic variable.

SWITCH Turn pseudo sense switches on or off.

*** Dataset Definition and Control ***

ACCESS Make a permanent dataset local.

ASSIGN Create a dataset and assign dataset characteristics.

HOLD Dataset release occurs with implicit HOLD.

NOHOLD Cancel effect of HOLD.

RELEASE Relinquish access to a dataset from a job.

*** Permanent Dataset Management ***

ACCESS Make a permanent dataset local.

ADJUST Redefine size of a permanent dataset.

DELETE Remove a permanent dataset.

MODIFY Change a permanent dataset's characteristic information.

NEWCHRG (DTRC) Change permanent file account number.

PERMIT Grant/deny access to a permanent dataset.

SAVE Make a dataset permanent.

SCRUBDS Write over a dataset before release.

*** Permanent Dataset Staging ***

See Chapter 3 for staging to and from the Mass Storage System.

ACQUIRE Get a front-end dataset and make it permanent.

DISPOSE Stage dataset to the front-end; release a local dataset; change disposition characteristics.

FETCH Get a front-end dataset and make it local.

MSACCES (DTRC) Supply your Username and password to the Mass Storage System (MSS).

MSFETCH (DTRC) Fetch a file from the MSS.

MSPURGE (DTRC) Purge a file from the MSS.

MSSTORE (DTRC) Store a file on the MSS.

SUBMIT Send local dataset to COS input queue.

*** Permanent Dataset Utilities ***

AUDIT Report on permanent datasets.

*** Local Dataset Utilities ***

BLOCK Convert an unblocked dataset to a blocked dataset.

COPYD Copy blocked datasets.

COPYF Copy blocked files.

COPYR Copy blocked records.

COPYU Copy unblocked datasets.

DS List local datasets.

NOTE Write text to a dataset.

QUERY Determine the current status and position of a local file.

REWIND Position a dataset at its beginning.

SKIPD Skip blocked datasets (position at EOD (after last EOF)).

SKIPF Skip blocked files from current position.

SKIPR Skip blocked records from the current position.

SKIPU Skip sectors on unblocked datasets.

UBBLOCK Convert a blocked dataset to an unblocked dataset.

WRITEDS Initialize a blocked dataset by writing a single file containing a specific number of records of a specific length.

*** Dumps and Other Aids ***

COMPARE Compare two datasets.

DEBUG Interpret a dump.

DUMPJOB Capture job information in dataset $DUMP for display by DUMP.

DUMP Display job information previously captured by DUMPJOB.

FLODUMP Dump flowtrace table.

FTREF Generate Fortran cross-reference.

ITEMIZE Report statistics about a library dataset.

PRINT Write value of JCL expression to the logfile.

SPY Generate a histogram of time usage within a program to locate inefficient code.

*** Logic Structure ***

ELSE IF-loop control.

ELSEIF IF-loop control.

ENDIF IF-loop termination.

ENDLOOP LOOP termination.

EXITIF IF-loop control.

EXITLOOP LOOP control.

IF Begin a conditional block of code.

LOOP Start of an iterative control statement block.

*** Procedures ***

See Section 2-3 for additional information on the creation of procedures.

CALL Transfer control to a procedure.

"call by name"
Execute a complex procedure in a library.

ENDPROC End of a procedure.

PROC Begin an in-line procedure definition block. This is followed by the procedure prototype statement which names the procedure and gives the formal parameter specifications.

RETURN Return control from a procedure to its CALLer.

*** Programming Languages ***

CFT Compile a Fortran source program.

CFT77 Alternate Fortran compiler (slower compile, faster execute).

PASCAL Compile a Pascal source program.

*** Program Libraries ***

See Section 2-4 for a discussion of program libraries (PL).

AUDPL	Audit an UPDATE PL.
UPDATE	Source and data maintenance.

*** Object Libraries ***

See Section 2-5 for a discussion of object libraries.

BUILD	Generate and maintain library datasets.
SEGLDR	Segment loader (see Section 2-6).

*** Miscellaneous ***

"call by name"	Execute a program by its local file name.
SID	Debug programs interactively or in batch.
SORT	Sort/merge.

*** JCL Expressions ***

An expression is a string of operands and operators. It is evaluated from left to right, taking into account parentheses and operator hierarchy. Expressions allow the incrementing of counters, error code checking, and string comparison.

There are four types of operands:

```
. integer constants   (+ddd... or -ddd... - decimal
                       nnn...B            - octal
                       range: 0 to ~10**19)

. literal constants   ('ccc...'L - left-justified,  zero-filled
                       'ccc...'R - right-justified, zero-filled
                       'ccc...'H - left-justified,  blank-filled
                       range of c: 040 - 176 octal
                       default: H)

. symbolic variables  (see below)

. subexpressions      (its value becomes an operand)
```

Expressions may be used in IF, ELSEIF, EXITIF, and EXITLOOP.

** Symbolic Variables **

There are 38 symbolic variables: 6 system constants, 7 variables set by COS, and 25 which can be set by the user.

* System Constants *

Symbol	Range	Description
FALSE	0	False
SID	literal	Mainframe ID (C1)
SYSID	literal	COS level ('COS n.nn')
TRUE	-1	True

SN and XM are also available.

* COS-set Variables *

Symbol	Range	Description
ABTCODE	0-nnn	COS job abort code (ABnnn)
DATE	literal	mm/dd/yy
FL	0-77777777	current octal field length
FLM	0-77777777	JOB statement maximum octal FL
PDMST	64-bits	status of most recent Permanent Dataset Manager request
TIME	literal	hh:mm:ss
TIMELEFT	64-bit integer	job time remaining (milliseconds)

* User-set Variables *

Symbol	Range	Description
G0-G7	64-bits	8 global pseudo-registers (can be used to pass data between procedures)
J0-J7	64-bits	8 job (local) pseudo-registers (each procedure level has its own J registers)
JSR	64-bits	Job Status Register containing the previous job step completion code
NOTEXT	64-bits	text field not echoed (default: ON)
PDMFC	64-bits	most recent user-issued PDM request
SSW1-SSW6	64-bits	pseudo sense switches

** Operators **

Operators may be

. arithmetic (+, -, *, /); Underflow and overflow are not detected; division by 0 produces zero

. relational (.EQ., .NE., .LT., .GT., .LE., .GE.); returns -1 (TRUE) or 0 (FALSE)

. logical (.OR., .AND., .XOR., .NOT.); returns a 64-bit value

Operations are performed left to right, taking into account parentheses, with the hierarchy of operators: (*, /), (+, -), relational, .NOT., .AND., .OR., .XOR..

** Strings **

A string is a group of ASCII characters (040-176 octal) to be taken literally. There are two types of strings:

. literal - delimited by apostrophes -- '...'

. parenthetical - delimited by parentheses -- (...)

Literal strings do not include the delimiters. An apostrophe within a literal string is represented by two apostrophes: '...''...'. A null string is indicated by two apostrophes: ''. A literal string is continued by placing an apostrophe and a continuation character at the end of the first line and an apostrophe at the start of the string on the next line:

```
...'This Is A '^
   'Long String.'    becomes    This Is A Long String.
```

Parenthetical strings do not include the delimiters. Spaces are removed; nested parentheses are not treated as separators; literal strings may appear in a parenthetical string. A parenthetical string is continued by placing a continuation character at the end of the first line and continuing the string on the next line:

```
        ...(This Is A ^
           Long String.)     becomes     ThisIsALongString.
```

***** Procedures *****

A procedure is a group of control statements separate from the job control statement dataset ($CS). Calling a procedure provides a simplified way to process that group of control statements. A procedure may be called by a job repeatedly or by another procedure.

There are two kinds of procedures in COS:

. simple - a sequence of control statements

. complex - a prototype statement (giving the name of the procedure and any parameters), the control statements, and optional data.

*** Simple Procedures ***

A simple procedure has no name or parameters and resides in a non-library dataset. It is invoked by a CALL without the CNS parameter. Control is returned to the caller by a RETURN statement, the end of the first file in the dataset, or an EXIT (when not skipping because of an error condition). A simple procedure has no parameter substitution.

Any COS JCL statement, except PROC and ENDPROC, may be used in a simple procedure. One use might be to access all the datasets needed in several jobs without having to specify them in the individual jobs.

*** Complex Procedures ***

Complex procedures are named and may have parameters described in a prototype statement. Complex procedures are executed by

. "call by name", which may include parameters for substitution in the procedure. The procedure is in $PROC or a local dataset named in a LIBRARY statement.

. CALL,DN=procfyl,CNS, followed by a line containing the procedure name and parameters for substitution. The procedure is the first file in a separate dataset; PROC and ENDPROC are not used.

Complex procedures may appear, delimited by PROC and ENDPROC, in the job control statement dataset ($CS). When PROC is encountered, the procedure is written to $PROC. Subsequent calls to the procedure may then be made using the procedure name (and any substitute parameters).

A complex procedure has the general form:

```
PROC.                          <-- not for CALL
prototype statement
control statements
...
&DATA,dn1.
data for first dataset
...
&DATA,dnn.
data for last dataset
ENDPROC.                       <-- not for CALL
```

** Prototype Statement **

The prototype statement defines the name of the procedure and its formal parameters with their default value(s). It has the form:

```
name,p1,p2,...,pn.
```

name - the name of the procedure (1-8 alphanumeric characters)

pi - a formal parameter specification

```
posi            - positional

keyi=dval:kval  - keyword
                  keyi   - formal keyword name
                  dval   - optional default value when keyi is
                           omitted from the calling statement
                  kval   - optional default value when keyi is
                           specified in the calling statement
                           without a value
keyi=           - no defaults; the caller must supply a non-null
                  value
keyi=:          - no defaults; allows keyi and keyi=
```

** Temporary Datasets **

One or more temporary datasets may be included in a complex procedure following the control statement. Each starts with

```
&DATA,dn.
```

where dn is the required dataset name.

** Parameter Substitution **

Formal parameters are used, preceded by an ampersand (&), within the body of the procedure. On execution, each is replaced by the value supplied or implied in the calling statement. &param is delimited by any character except A-Z, a-z, 0-9, @, $, or %. If the next character is one of these, the underline (_) is used as the delimiter and is removed at execution time.

If too few positional parameters are specified by the caller, null strings are used for the remaining parameters; if too many, the job aborts. Keyword parameters may appear in any order, however, all positional parameters must precede all keywords.

** Apostrophes and Parentheses **

Apostrophes in the calling statement denote literals and are not removed during substitutions; the outer set of parentheses are removed. If you are not sure how a parameter is used in the procedure, enclose it in parentheses.

The following shows parenthetical substitution:

caller	after substitution
value	value
(value1=value2)	value1=value2
value1'.'value2	value1'.'value2
value1(.)value2	value1.value2

*** DTRC Procedure Library ***

One procedure library has been added to COS at DTRC:

```
PROCLIB,OWN=PUBLIC.
```

```
To use:  ACCESS,PROCLIB,OWN=PUBLIC.
         LIBRARY,PROCLIB:*.
         procname,....
```

*** Examples ***

** Simple Procedures **

1) The first file of dataset GETLIBS contains:

```
ACCESS,DN=MSPROC,OWN=PUBLIC.      <-- the MSS procedures
ACCESS,DN=DTLIB,OWN=PUBLIC.       <-- the DTLIB subroutine library
ACCESS,DN=SUBS.                   <-- your subroutine library
```

This is executed by:

```
CALL,DN=getlibs.
```

** Complex Procedures **

2) As in example 1, but your subroutine library is to be identified by the caller:

```
GETLIBS,SUBS.                     <-- prototype statement
ACCESS,DN=MSPROC,OWN=PUBLIC.      <-- the MSS procedures
ACCESS,DN=DTLIB,OWN=PUBLIC.       <-- the DTLIB subroutine library
ACCESS,DN=SUBS,PDN=&SUBS.         <-- your subroutine library
```

When called by:

```
CALL,DN=getlibs,CNS.
getlibs,othersubs.
```

the third ACCESS expands to ACCESS,DN=SUBS,PDN=othersubs. Note that the name of the procedure is unimportant, since it is the only procedure in the file. "getlibs,othersubs." could be replaced by "*,othersubs".

When called by:

```
CALL,DN=getlibs,CNS.
getlibs,(hislib,OWN=him).
```

the third ACCESS expands to ACCESS,DN=SUBS,PDN=hislib,OWN=him.

When called by:

```
CALL,DN=getlibs,CNS.
getlibs,'hislib,OWN=him'.
```

the third ACCESS expands to ACCESS,DN=SUBS,PDN='hislib,OWN=him'. While this is legal (it says the permanent filename is "hislib,OWN=him"), it is probably an error and, if so, will abort the procedure.

3) Create a procedure library from procedures in the job stream.

```
...
ECHO,OFF.
RELEASE,DN=$PROC.          <-- return existing $PROC
*
PROC.                      <-- write first procedure to $PROC
prototype
procedure body
RETURN...procname
EXIT.
RETURN,ABORT...procname
ENDPROC.                   <-- end of first procedure
*
PROC.                      <-- write next procedure to $PROC
prototype
procedure body
RETURN...procname
EXIT.
RETURN,ABORT...procname
ENDPROC.                   <-- end of procedure
*
...                        <-- more procedures
*
ACCESS,DN=proclib,NA,UQ.   <-- get original (existing) library
SAVE,DN=$PROC,PDN=proclib. <-- save new library
DELETE,DN=proclib,NA.      <-- delete original library
RELEASE,DN=$PROC.          <-- return new library
ACCESS,DN=proclib.         <-- get new library with its own name
LIBRARY,DN=*:proclib.      <-- add it to the end of the library
                               list
-or-
LIBRARY,DN=proclib:*       <-- add it to the beginning of the
                               library list
ECHO,ON.
...
< use one of the procedures >
...
```

4) Create a procedure library from procedures in a separate file.

```
    ...
    FETCH,DN=myprocs,TEXT='myprocs.pro'.       <-- defaults to AC=ST
    CALL,DN=myprocs.
    SAVE,DN=$PROC,PDN=proclib,PAM=R.           <-- others may use it
    ...
```

where VMS file MYPROCS.PRO contains:

```
    *            first procedure
    PROC.
    prototype
    procedure body
    ENDPROC.
    *            next procedure
    PROC.
    prototype
    procedure body
    ENDPROC.
    *            next procedure
    ...                           <-- more procedures
    *
```

***** Program Libraries *****

Source programs and data may be in separate datasets or may be stored and maintained in program libraries. UPDATE creates and maintains these libraries while AUDPL (see Appendix B) audits them.

*** UPDATE ***

UPDATE is a program for creating and modifying a program library (PL). In addition, UPDATE will extract individual modules for input to a compiler or other program.

By default, 72 columns of information are retained. Fifteen additional characters are retained for each line: an 8-character identifier, a period (.), and a 6-digit sequence number, i.e., id.seq.

UPDATE supports two kinds of text modules or decks:

- a regular deck (beginning with a DECK directive)
- a common deck (beginning with a COMDECK directive) which may be included in decks with a CALL directive

Each type includes all lines following the deck directive until the next deck or modification directive.

History information is retained allowing the deletion, modification, or restoration or previous modifications.

See Appendix B for a description of the UPDATE control statement parameters.

*** UPDATE Directives ***

An UPDATE directive, which must be in upper case, has the following format:

```
m directive_name  [ parameters ]
```

where m is the master character (default: asterisk (*)). There are five categories of directives.

** DECK and COMDECK **

*DECK deck (*DK)
First line of a new deck. <deck> is up to 8 characters, any ASCII character from 41 to 176 octal, except comma, period, blank, colon, equals.

*COMDECK cmdk (*CDK)
First line of a new common deck.

** Compile Directives **

```
*CALL cmdk        (*CA)
             Include the contents of a common deck.

*CWEOF
             Write an EOF on the compile dataset if anything was
             written since the last EOF.

*NOSEQ
             Do not write sequence numbers.

*SEQ
             Write sequence numbers.

*WEOF
             Write an EOF on the compile dataset.

*WIDTH dw
             Change the data width (default: 72).
```

*IF, *ELSEIF, *ELSE, and *ENDIF are also available.

** Modification Directives **

```
*BEFORE id.seq     (*B)
             Insert before a line.

*COPY p,id1.seq1,id2.seq2     (*CY)
             Copy a range of lines from deck or comdeck <p>.

*DELETE id1.seq1              (*D)     <-- one line
*DELETE id1.seq1,id2.seq2              <-- a range of lines
*DELETE id1.seq1,.seq2                 <-- same (short form)
             Delete a line or a range of lines.

*IDENT ident     (*ID)
*IDENT ident,K=k1:k2:...,U=u1:u2:...
             Identify a set of modifications.  You can require that
             other modification sets be known (K=) or unknown (U=).

*INSERT id.seq     (*I)
             Insert after a line.

*RESTORE id1.seq1              (*R)     <-- one line
*RESTORE id1.seq1,id2.seq2              <-- a range of lines
*RESTORE id1.seq1,.seq2                 <-- same (short form)
             Restore a line or a range of lines.
```

** Run Options **

*/comment
A comment line.

*COMPILE p1,p2,...,pj.pk,...,pn (*C)
Write one or more decks, including a range (pj.pk), to the compile and/or source datasets. Use UPDATE,K to force the output order.

*COPY p,id1.seq1,id2.seq2,dn (*CY)
*COPY p,id1.seq1,id2.seq2,dn,SEQ
Copy a range of lines from deck or comdeck <p> to dataset <dn>. SEQ will include sequence numbers.

*LIST
Resume listing input lines. UPDATE,L=0 overrides *LIST.

*MASTER m
Define a new master character for subsequent directives. (default: *)

*NOLIST
Stop listing input lines. *NOLIST overrides UPDATE,IN.

*READ dn (*RD)
Read input from another dataset.

*REWIND dn
Rewind a dataset.

*SKIPF dn
*SKIPF dn,n
Skip file(s) in a local dataset.

*DECLARE and *DEFINE are also available.

** Input Edit Directives **

*EDIT p1,p2,...,pn (*ED)
Remove deleted and yanked lines from specific decks. These lines cannot be retrieved. This is useful for cleaning up a PL.

*MOVEDK dk1:dk2
*MOVEDK dk1:.
Position deck of common deck <dk1> immediately after deck or common deck <dk2> or at the beginning of the PL <.>.

*PURGE id1,id2,...,idj.idk,...,idn..
Remove the effect of a modification set (idi), a range of datasets (idj.idk), or a set and all following (idn..).

*PURGEDK dk
Permanently remove a deck or common deck.

*UNYANK id1,id2,...,idj.idk,...,idn..
Reactivate a deck, comdeck, or modification set previously yanked.

*YANK id1,id2,...,idj.idk,...,idn..
Temporarily delete a deck, comdeck, or modification set previously yanked.

*SKIP and *ENDSKIP are also available.

*** Examples ***

1) Create a PL:

```
JOB,JN=makepl1.
ACCOUNT,....
UPDATE,P=0,C=0.                  <-- no $PL or $CPL
SAVE,DN=$NPL,PDN=mypl.
/EOF
*DK DECK1
   <lines for deck DECK1>
*DK DECK2
   <lines for deck DECK2>
*DK DECK3
   <lines for deck DECK3>
```

2) Extract, compile and execute deck DECK2 from PL MYPL:

```
JOB,JN=getpl2.
ACCOUNT,....
ACCESS,DN=$PL,PDN=mypl.
UPDATE.
CFT,I=$CPL.
SEGLDR,CMD='MAP,PART',GO.
/EOF
*C DECK2
```

3) Create a PL using a common deck, compile and execute:

```
JOB,JN=makepl3.
ACCOUNT,....
UPDATE,P=0.                      <-- no $PL (required to create)
SAVE,DN=$NPL,PDN=mypl.
CFT,I=$CPL.
SEGLDR,CMD='MAP,PART',GO.
/EOF
*CDK COM3
      common / mycom / a, b, c
      real a, b, c
*DK PROG3
      program prog3
*CALL COM3
      call sub
      print *, 'a,b,c=', a, b, c
      end
*DECK SUB
      subroutine sub
*CA COM3
      a = 1.
      b = 2.
      c = 3.
      return
      end
```

4) Update old source library to new, compile all decks and execute:

```
JOB,JN=job4.
ACCOUNT,....
ACCESS,DN=$PL,PDN=mylib.
UPDATE,F,N.
SAVE,DN=$NPL,PDN=mylib.
CFT,I=$CPL.
SEGLDR,GO.
/EOF
*IDENT DS0620          <-- correction must be unique (initials,date)
*INSERT ALONE.57       <-- correct deck ALONE by insert after line 57
   <FORTRAN statements>
*DELETE FOUR.12,13     <-- correct deck FOUR replacing lines 12-13
   <new lines to replace deletions - optional>
/EOF
   <data lines, if any>
/EOF
```

5) Select routines from source subroutine library on MSS and compile with own program:

```
JOB,JN=job5.
ACCOUNT,....
ACCESS,DN=MSPROC,OWN=PUBLIC.
LIBRARY,DN=MSPROC:*.
MSACCES,UN=un,MPW=mymsspw.
CFT.                    <-- compile own programs
MSFETCH,DN=LIBR,MDN=DTLIBPC,UN=NSYS.
UPDATE,P=LIBR,Q,L=0.
CFT,I=$CPL,L=0.         <-- omit L=0 to get listing
SEGLDR,GO.              <-- load and execute
/EOF
    <own FORTRAN decks>
/EOF
*C rtn1,rtn6.rtn8       <-- select decks RTN1, 6, 7, 8 from library
/EOF
    <data records, if any>
/EOF
```

***** Object Libraries *****

BUILD is a utility for creating and maintaining libraries of absolute and relocatable object modules. These libraries can then be used by the loader to locate the program to execute or the subprograms to be loaded with your program.

The BUILD control statement and BUILD directives are described in Appendix B.

*** DTRC Object Libraries ***

Two object libraries have been added to COS at DTRC:

```
DTLIB,OWN=PUBLIC    - Subprograms written or maintained by the
                      Computer Center
                      To use: ACCESS,DN=DTLIB,OWN=PUBLIC.
                              SEGLDR directive: LIB=DTLIB

UTILITY,OWN=PUBLIC  - Programs written or maintained by the
                      Computer Center
                      To use: ACCESS,DN=UTILITY,OWN=PUBLIC.
                              LIBRARY,UTILITY:*.
                              program_name,....
```

*** Examples ***

1) Create a library of subprograms:

```
JOB,JN=JOB1.
ACCOUNT,....
CFT.
BUILD,I=0,OBL=0.
SAVE,DN=$NBL,PDN=MYSUBLIB.
/EOF
   <Fortran source subprograms>
/EOF
```

2) Create a library of all subprograms from an UPDATE library:

```
JOB,JN=JOB2.
ACCOUNT,....
ACCESS,DN=$PL,PDN=MYPL.
UPDATE,F.
CFT,I=$CPL.
BUILD,I=0,OBL=0.
SAVE,DN=$NBL,PDN=MYSUBLIB.
/EOF
```

3) Add a subprogram to an existing library and have the output list in alphabetical order.

```
JOB,JN=JOB3.
ACCOUNT,....
ACCESS,DN=$OBL,PDN=MYSUBLIB.
CFT.
BUILD,I=0,SORT.
SAVE,DN=$NBL,PDN=MYSUBLIB.
/EOF
   <Fortran source subprograms>
/EOF
```

4) Delete subprogram BADSUB from an existing library and list the contents of both old and new libraries.

```
JOB,JN=JOB4.
ACCOUNT,....
ACCESS,DN=$OBL,PDN=MYSUBLIB.
BUILD,B=0.
SAVE,DN=$NBL,PDN=MYSUBLIB.
/EOF
OMIT BADSUB
LIST
```

5) List the contents of an existing library.

```
JOB,JN=JOB5.
ACCOUNT,....
ACCESS,DN=SUBLIB,PDN=MYSUBLIB.
BUILD,OBL=0,NBL=0,B=0.
/EOF
FROM SUBLIB; LIST.
```

***** Loader *****

The loader is responsible for loading all programs, resolving any external references, and optionally initiating execution. Loading can produce either a single absolute module, or a (segmented) absolute program in which different parts of a program reside in memory only when needed.

*** SEGLDR ***

The primary loader is SEGLDR. It is controlled by directives which may appear as the next file in the input stream, in a separate file, or in the loader control statement.

** Control Statement **

See Appendix B for a fuller description of the SEGLDR control statement.

```
SEGLDR,I=dirfile,L=listfile,DW=dirwidth,CMD='directives',GO.
```

"SEGLDR." implies SEGLDR,I=$IN,L=$OUT,DW=80.

** Message Levels **

SEGLDR issues messages at the following levels:

ERROR - immediately terminates SEGLDR with no executable output

WARNING - no executable output but processing continues

CAUTION - executable output but a possible error was found

NOTE - SEGLDR has been misused or used ineffectively; executable output is still valid

COMMENT - does not affect execution

** Directives **

Most SEGLDR directives have the format: keyword=value. Comments (anything following an asterisk (*)) may appear anywhere in the directives, including at the end of a directive line. Multiple comments on a line are separated by a semicolon (;). Elements of a list are comma-separated. Directives may be continued by splitting the line after a parameter (the comma is the last non-blank character in the line).

Naming files: ABS, BIN, LIB, NODEFLIB.

Listing control: COMMENT, ECHO, MAP, TITLE, TRIAL.

Naming modules and common blocks: COMMONS, DYNAMIC, FORCE, MODULES.

Error message control: DUPENTRY, MLEVEL, REDEF, USX.

Entry point and execution control: EQUIV, SET, XFER.

Global heap memory management: HEAP, LOWHEAP, STACK.

Memory allocation and presetting: ALIGN, ORG, PRESET.

Symbolic debugging: SID, SYMBOLS.

Miscellaneous COS-dependent directives: ABORT, BCINC, GRANT, NOECHO,
NORED, PADINC, SECURE.

Miscellaneous GLOBAL DIRECTIVES: CASE, CPUCHECK.

Additional information, including directives not discussed here, may be found in SR-0066 Segment Loader Reference Manual.

* comment A comment.

Examples:

```
TITLE=GLOBAL DIRECTIVES
*----------------------
* Global directives
*----------------------
BIN=ABC
TITLE=TREE DIRECTIVES
*--------------------
* Tree directives
*--------------------
TREE
  ROOT(A,B)
ENDTREE
TITLE=SEGMENTS
*-------------
SEGMENT=ROOT
* ROOT directives
```

ABORT=ON ¦ OFF

Control SEGLDR error termination.

Values: ON - abort if errors
OFF - terminate normally even if errors

Default: ABORT=ON

ABS=dn

The dataset to contain the absolute module.

Default: $ABD

Examples: ABS=myprog

ALIGN=IGNORE ¦ MODULES ¦ NORMAL

Control the starting locations of modules and common blocks.

Values:

IGNORE - start each module's local or common block at the word following the previous one (ignore align bit)

MODULES - start each module's local block and common block (if the align bit is set) at an instruction buffer boundary (32 words)

NORMAL - start each module's local or common block with the align bit set at an instruction buffer boundary (32 words)

Default: ALIGN=NORMAL

BIN=dn1,dn2,...

Datasets containing the relocatable modules to be loaded.

Default: BIN=$BLD

Examples:

```
BIN=myfile,yourfile,
    theirfyl
BIN=oldfile
```

CASE=UPPER ¦ MIXED

Control character conversion in directives.

Values: UPPER - convert to upper case
MIXED - do not convert

Default: CASE=UPPER

```
COMMONS=blk1:siz1,blk2:siz2,...
             Specify the order to load common blocks.

             Values:    blki - name of a common block

                        sizi - n - decimal size
                               0 - first occurrence of this block sets
                                   the size
                               (default: 0)

             Examples:  COMMONS=myblk:100000,data1
                                  ^-- MYBLK is 100,000 words (no
                                      matter how it is defined); DATA1
                                      has its first encountered length


DUPENTRY=ERROR ¦ WARNING ¦ CAUTION ¦ NOTE ¦ COMMENT ¦ IGNORE
             Specify the message level for a duplicate entry point.

             Default:   DUPENTRY=CAUTION


DYNAMIC=comblk
DYNAMIC=//
            Name a common block to be located after the largest segment
            or the heap (if required).  You control its size.  It is
            always available to the program and cannot be preloaded
            with data.

             Values:    a COMMON block name or // (blank common)

             Default:   no dynamic common blocks

             Examples:  DYNAMIC=ARRAYS
                                  ^-- common block /ARRAYS/ is dynamic


ECHO=ON ¦ OFF
             Resume or suppress listing of input directives.

             Default:   ECHO=OFF


EQUIV=epname(syn1,syn2,...)
             Substitute a call to one entry point for a call to
             another.

             Values:    epname - the entry point to be used in the
                                 substitution

                        syni   - an entry point to be replaced by
                                 epname
```

```
                Examples:   ...
                          CALL A
                            ...
                          CALL B
                            ...

                          EQUIV=C(A,B)
                                ^-- replaces the calls to A and B
                                    by calls to C


FORCE=ON ¦ OFF
                Control the forced loading of modules whose entry points
                are never called.

                Default:  FORCE=OFF


LIB=lib1,lib2,...
                Libraries to be searched for routine not included in BIN-
                files.

                Examples: ACCESS,DTLIB,OWN=NSYS.      <-- DTRC subroutine
                                                          library
                          ACCESS,sublib.              <-- your subroutine
                                                          library
                          SEGLDR,CMD='LIB=sublib,DTLIB',...


MAP=NONE ¦ STAT ¦ ALPHA ¦ ADDRESS ¦ PART ¦ EPXRF ¦ CBXRF ¦ FULL
                Control the map listing.

                Values:   NONE    - no map
                          STAT    - list load statistics: date/time,
                                    longest branch length, last segment,
                                    transfer entry point, stack and heap
                                    information
                          ALPHA   - STAT + block map for each segment
                                    (modules in alphabetical order)
                          ADDRESS - ALPHA but modules in address order
                          PART    - ALPHA + ADDRESS
                          EPXRF   - STAT + entry point cross reference
                          CBXRF   - STAT + common block cross reference
                          FULL    - PART + EPXRF + CBXRF

                Default:  MAP=NONE

                Examples: MAP=STAT
                          MAP=EPXRF,CBXRF
```

```
MLEVEL=ERROR ¦ WARNING ¦ CAUTION ¦ NOTE ¦ COMMENT
             Print messages down to and including the level specified
             (has no effect if L=0).

             Default:   MLEVEL=CAUTION

             Examples:  MLEVEL=NOTE
                                   ^-- print error, warning, caution,
                                       and note messages


MODULES=mod1:ds1,mod2:ds2,...
             The modules to be included and, optionally, the dataset
             containing a specific module.

             Values:    modi - name of module to be loaded

                        dsi  - optional dataset containing the module

             Examples:  MODULES=sub1:sublib,sub2,sub3:yourlib
                        MODULES=sub4,sub5
                                   ^-- get SUB1 from SUBLIB; SUB3 from
                                       YOURLIB; SUB2, SUB4, SUB5 from
                                       the first dataset containing them


NODEFLIB     Do not search the default libraries.  Search only BIN and
             LIB datasets.

             NOTE: Segmented loads must specify the file containing
                   routine $SEGRES.

             Examples:  NODEFLIB; LIB=sublib,DTLIB,$SCILIB


ORDER=MODULES,COMMONS ¦ COMMONS,MODULES ¦ XMP.EMA
             Load modules before or after commons.

             Values:    XMP.EMA - most efficient allocation on X-MP
                                  having more than 4 million words

             Defaults:  ORDER=MODULES,COMMONS (<=4 million words)
                        ORDER=XMP.EMA         (> 4 million words)


PRESET=ONES ¦ ZEROS ¦ INDEF ¦ -INDEF ¦ value
             Preset uninitialized data areas.

             Values:    ONES   - set to -1
                        ZEROS  - set to 0
                        INDEF  - set to octal 0605054000000000000000
                        -INDEF - set to octal 1605054000000000000000
                        value  - 16-bit value placed in each parcel
                                 (0 < value < 177777 octal)

             Default:   PRESET=ZEROS
```

SID — Load for debugging. Symbol tables are written to $DEBUG (or SYMBOLS=dn).

Default: Normal load

SYMBOLS=ON ¦ OFF ¦ dn
Specify program symbol table handling.

Values:
ON - write symbol table to $DEBUG
OFF - ignore symbol table
dn - write symbol table to dn
(dn may not be ON or OFF)

Default: SYMBOLS=ON

TITLE=title
TITLE — Define the second line of each page header. A page eject is forced.

Value:
title - a string of 0-74 characters
(ends with end-of-line or semicolon)
omitted - clear the second header line

Examples: TITLE=This is a user title, really!

TRIAL — Do not generate an executable module. Lets you get the load map, determine optimal memory usage for data, or get the total memory required.

Examples: TRIAL

USX=WARNING ¦ CAUTION ¦ IGNORE
Specify how to treat unsatisfied externals.

Values:
WARNING - issue a warning message;
do NOT write executable output
CAUTION - issue a caution message;
write executable output
IGNORE - issue no message;
write executable output

Default: USX=CAUTION

*** Segmentation ***

To make a large program fit into memory, it may be structured in segments, so that only a portion of the program resides in memory. By using the tree structure directives of SEGLDR, different arrangements of a program can be tried, without changing the program, until the best is achieved.

** Segmentation Directives **

Tree definition: TREE, tree_definition, ENDTREE.

Segment description: SEGMENT, BIN, COMMENT, COMMONS, DUP, ECHO, MODULES, SAVE, TITLE, ENDSEG.

Global: COPY, SAVE, SLT.

```
BIN=dn1,dn2,...
             Datasets containing the relocatable modules to be loaded.
             Only the first file of each dataset is processed.

             Default:   BIN=$BLD

             Examples:  SEGMENT=birch
                          BIN=myfile,yourfile,
                              theirfyl
                          BIN=oldfile
                        ENDSEG
                                   ^-- all modules in datasets MYFILE,
                                       YOURFILE, THEIRFYL, and OLDFILE
                                       are loaded into segment BIRCH
```

```
COMMONS=blk1:siz1,blk2:siz2,...
             Specify the order to load common blocks.

             Values:    blki - name of a common block

                        sizi - n - decimal size
                               0 - firnt occurrence of this block sets
                                   the size
                               (default: 0)

             Examples:  COMMONS=myblk:100000,data1
                                   ^-- MYBLK is allocated 100,000 words
                                       (no matter how it is defined);
                                       DATA1 has its first encountered
                                       length
```

COPY

Force the program to execute from a scratch file. This may speed program execution, especially of programs with segments which are loaded many times, because a faster form of I/O is used. SAVE=ON also forces the use of a scratch file.

Default: a scratch file is not used

DUP=mod(seg1,seg2,...)

Specify that a module is to be loaded into several segments. DUP must appear before the definitions of the segments into which the module is to be placed.

An alternate way is to list the module in the MODULES or COMMONS directive of each segment.

```
Examples:  DUP=sub3(seg1,seg2)                      root
           SEGMENT=seg1                               |
             MODULES=sub1                 ____________|____________
             COMMONS=com1                 |                       |
           ENDSEG                        seg1                    seg2
           SEGMENT=seg2
             MODULES=sub2                com1                    com1
             COMMONS=com1                sub1                    sub2
           ENDSEG                        SUB3                    SUB3
```

ENDSEG

End the definition of a segment of a tree structure.

Examples: see SEGMENT

ENDTREE

End the definition of a tree structure.

Examples: see TREE

MODULES=mod1,mod2,...

(segment) List the modules to be put into the segment.

Values: modi - module name and optional dataset from which it is to be loaded (mod:ds)

```
Examples:  MODULES=m:binm,n,o
                     ^-- load module M from dataset BINM
                         and modules N and O from the
                         first dataset which contains
                         them
```

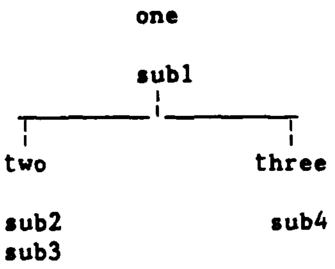
SAVE=ON ¦ OFF

(Global) Specify whether all segments are to be saved (written to disk) before being overlaid. SAVE in a segment overrides the global SAVE.

```
Values:    OFF - do not save each segment
           ON  - save each segment

Default:   SAVE=OFF

Examples:  SAVE=ON
           TREE
             one(two,three)
           ENDTREE
           SEGMENT=one
             MODULES=sub1
           SEGMENT=two
             MODULES=sub2,sub3
           SEGMENT=three
             SAVE=OFF
             MODULES=sub4
           ENDSEG
```



SEGMENT=segname

Begin the description of the contents of one segment of a tree.

```
Examples:  SEGMENT=oak
             MODULES=k,l,m
             COMMONS=//,oakcom
           ENDSEG
```

TREE

End the global directives and start the definition of a tree structure.

```
Examples:  TREE
             tree structure
           ENDTREE
```

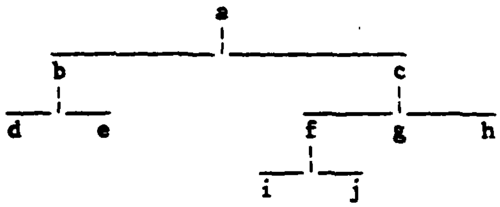
tree segment structure

Define the tree structure, that is, the segments in each branch of the tree. The order of these definitions is unimportant.

Syntax: segname(seg1,seg2,...)

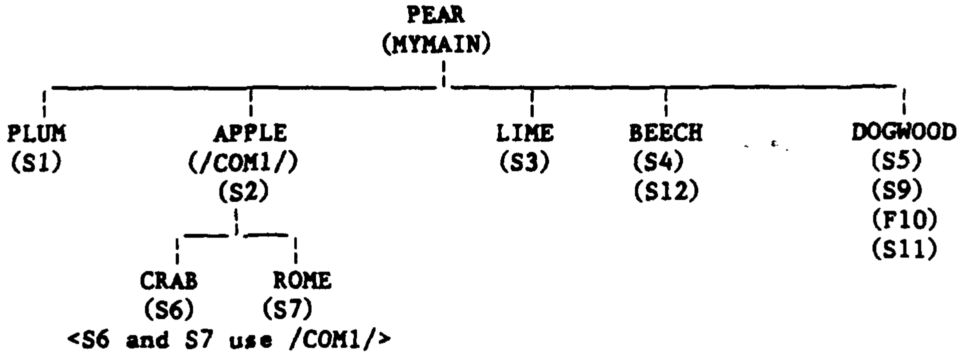
Examples:

```
TREE
  a(b,c)
  b(d,e)
  c(f,g,h)
  f(i,j)
ENDTREE
```



** Sample Tree Diagram **

A block data subprogram defines common /COM1/ which is to be loaded with program S2. /COM1/ is also referred to by S6 and S7.



```
TREE
  pear(plum,apple,lime,beech,dogwood)
  apple(crab,rome)
ENDTREE
SEGMENT=pear
  MODULES=mymain
SEGMENT=plum
  MODULES=s1
SEGMENT=apple
  MODULES=s2
  COMMONS=com1
SEGMENT=line
  MODULES=s3
SEGMENT=beech
  MODULES=s4,s12
SEGMENT=dogwood
  MODULES=s5,s9,f10,s11
SEGMENT=crab
  MODULES=s6
SEGMENT=rome
  MODULES=s7
ENDSEG
```

** Segmentation Cautions **

1. To develop a segmented job, several runs may be required, so relocatable object code should be SAVEd. Common blocks and some system routines may need to be included in lower segments to operate properly.

2. The load map should be checked carefully for any duplicate common block entries. The same common block may appear in more than one segment, each being considered a different common block. References are to the common block in the segment, if none, then to the one on the same branch. If a given common block is to appear only once in a program (the normal case), then it should be placed in the segment nearest to the root segment which can be referenced by all segments which use it.

*** Compile, Load and Save an Absolute Program ***

** Simple Load **

```
JOB,JN=jobname,....
ACCOUNT,....
CFT.
SEGLDR,CMD='ABS=myprog'.
SAVE,DN=myprog,PAM=R.          <-- read only
/EOF
      PROGRAM MYPROG (...
          ...
/EOF
```

** Segmented LOAD **

```
JOB,JN=jobname,....
ACCOUNT,....
CFT.
SAVE,DN=$BLD,PDN=myprogob.        <-- save relocatable modules for
                                      possible re-segmentation
SEGLDR.
SAVE,DN=myprog,PAM=R.             <-- read only
/EOF
   (CFT source program)
/EOF
ABS=myprog
TREE
  ...
ENDTREE
SEGMENT=...
  ...
ENDSEG
SEGMENT=...
  ...
ENDSEG
  ...
/EOF
```

***** The Mass Storage System *****

The Mass Storage System (MSS) is a large capacity on-line mass storage device. It is a cost effective extension to the Cray, CDC and VAXcluster disk systems and conventional magnetic tape storage. Specifically, the MSS, which is part of the CDC CYBER 860A, offers:

- More than 20 times the on-line storage of the VAXcluster system; more than 40 times the on-line storage of the Cray X-MP.
- On-line access to files which previously had to be stored on magnetic tape because of size restrictions and/or infrequent use.
- Reduced storage charges for these on-line files.

*** MSS Security ***

To provide adequate security for MSS users, you must submit your MSS (CYBER 860) password in any non-CDC job or interactive session which will manipulate MSS files. To protect your MSS files, you must change this password at least every 90 days using the PASSWOR command on the CDC CYBER 860A or the HFT PASSWORD command on the VAXcluster.

*** MSS File Purge ***

MSS files may be purged by the Computer Center if the job order number is invalid or has been cancelled.

To recover purged files, call User Services, Code 1893.1, (202) 227-1907. A nominal fee will be charged for this service. After the files have been restored, you must change to your valid job order number (on 860: CHANGE,pfn/CP or BEGIN,NEWCHRG).

*** MSS Backup for Critical Files ***

In addition to normal file backup, critical direct files may be backed up and stored off-station. These files are available in the event of a catastrophe (such as fire) at the Carderock Computer Center.

For a file to be designated as "critical", it must have the attribute Backup Requirement (BR) set to critical (CR). This is done by specifying "BR=CR" if the file is critical, or "BR=Y" if it is not, when the file is made permanent. The default is BR=Y meaning on-station backup. For example:

```
DEFINE,lfn=mfn/BR=CR.    <-- store a critical file
CHANGE,mfn/BR=CR.        <-- make a file critical
CHANGE,mfn/BR=Y.         <-- make a file non-critical
```

Files designated for this off-station backup service will be charged a higher rate.

Because an indirect file is just part of a larger file (page 5-1-6) that may contain several users' files, it cannot be designated as critical.

*** Using the MSS from the Cray ***

A description of the syntax of these commands may be found in Appendix B.

ACQUIRE Transfer a file from the MSS as a local dataset and make it permanent on the Cray.

```
Examples:  ACQUIRE,DN=SOURCE,SDN=MYFILE,PDN=MYFILE,MF=N1,^
                   TEXT='USER,user,pw.ATTACH,MYFILE.CTASK.'.
                          ^-- transfer your direct MSS file
                              MYFILE as local dataset SOURCE
                              and make it a permanent dataset
                              named MYFILE

           ACQUIRE,DN=DATA46,PDN=DATA46,MF=N1,^
                   TEXT='USER,user,pw.'^
                   'ATTACH,DATA46/UN=ABCD,PW=filepw.CTASK.'.
                          ^-- transfer user ABCD's MSS file
                              DATA46 (assuming you have
                              permission to read the file) as
                              local dataset DATA and make it
                              a permanent dataset named DATA46
```

DISPOSE Transfer a Cray local dataset to the MSS.

```
Examples:  DISPOSE,DN=FT13,MF=N1,SDN=MYOUT13,DC=ST,^
                   TEXT='USER,user,pw.'^
                        'PURGE,MYOUT13/NA.'^
                        'DEFINE,MYOUT13.'^
                        'CTASK.'.
                          ^-- local dataset FT13 is
                              transferred to the MSS where it
                              will be known as MYOUT13
```

FETCH Transfer a file from the MSS as a local dataset. It is released at the end of the job.

```
Examples:  FETCH,DN=SOURCE,SDN=MYFILE,MF=N1,^
                 TEXT='USER,user,pw.ATTACH,MYFILE.CTASK.'.
                          ^-- transfer your MSS file MYFILE as
                              local dataset SOURCE

           FETCH,DN=ABDATA,MF=N1,TEXT='USER,user,pw.'^
                 'ATTACH,ABDATA/UN=ABCD,PW=filepw.CTASK.'.
                          ^-- transfer user ABCD's MSS file
                              ABDATA as local dataset ABDATA

           FETCH,DN=SOURCE,SDN=MYFILE,MF=N1,^
                 TEXT='USER,user,pw.GET,MYINDF.CTASK.'.
                          ^-- transfer your CYBER 860 indirect
                              file MYFILE as local dataset
                              SOURCE
```

The following procedures simulate the MSS commands of the CDC NOS/BE system at DTRC. To use them, you must first

```
ACCESS,DN=PROCLIB,OWN=PUBLIC.  <-- access the procedure library
LIBRARY,DN=PROCLIB:*.          <-- add to your library set
```

MSACCES Supply your Username and password to the MSS. MSACCES is required before you can MSFETCH, MSPURGE or MSSTORE.

```
Examples:  MSACCES,US=myid,MPW=mymsspw.
```

MSFETCH Fetch a direct file from the MSS.

```
Examples:  MSFETCH,DN=infyl,MDN=mydata.
                          ^-- your file in transparent mode

           MSFETCH,DN=prog,MDN=othrpgm,UN=ABCD,PW=pgmpw.
                          ^-- another user's file
```

MSPURGE Purge an MSS file.

```
Example:   MSPURGE,DN=myfyle.
```

MSSTORE Store a file on the MSS as a direct file.

```
Examples:  MSSTORE,DN=out1,MDN=outfyl1,NA=1.
                          ^-- overwrite if file already exists

           MSSTORE,DN=fyl2,MDN=file2,DF=CB.
                          ^-- file is stored in CDC Display
                              Code
```

*** Using the MSS from the VAXcluster ***

A description of the syntax of these commands may be found by typing "HELP HFT" on the VAXcluster.

```
HFT     HYPERchannel (direct) File Transfer.

        Examples: HFT ACCESS /U=ABCD /A=1222233344 /P=MSS_password
                                 ^-- gain access to the MSS

                  HFT CHANGE "MYFILE/AC=newac,CT=PU"
                                 ^-- change account number of MSS
                                     file MYFILE and make it public

                  HFT DEFAULT
                                 ^-- display your current ACCESS
                                     values

                  HFT DELETE MYFILE
                                 ^-- delete MSS file MYFILE

                  HFT DIRECTORY
                                 ^-- audit your MSS file names

                  HFT DIRECTORY "LO=F"
                                 ^-- full audit of your MSS files

                  HFT FETCH MYPROG MYPROG.FOR
                                 ^-- fetch your MSS file MYPROG and
                                     make it permanent file
                                     MYPROG.FOR

                  HFT PASSWORD
                  old password
                  new password
                  new password repeated
                                 ^-- change your MSS password

                  HFT PERMIT "MYFILE/UN=xxxx,M=R"
                                 ^-- give read access to user xxxx

                  HFT STORE MYPROG.FOR "MYPROG/CT=S"
                  HFT STORE MYPROG.FOR "MYPROG/CT=S" /DELETE
                                 ^-- store your file MYPROG.FOR on the
                                     MSS as MYPROG (/DELETE will
                                     delete your VAXcluster permanent
                                     file)
```

```
MSSAUDIT Audit your MSS files in a variety of formats.

        Examples:  MSSAUDIT  S    <-- get a sorted short audit of
                                      your MSS files at the terminal

                   MSSAUDIT  F  MSSAUDIT.LIS
                                  ^-- put a sorted full audit of your
                                      MSS files into file MSSAUDIT.LIS

                   MSSAUDIT  0  UN=xxxx
                                  ^-- display a sorted list of the MSS
                                      files owned by user xxxx
                                      (assuming you have permission to
                                      see them)


MSSBACKUP Store several files in a single file on the MSS, retaining
          each file's characteristics.  Fetch individual files from the
          MSS file previously stored by MSSBACKUP.

          Examples:  MSSBACKUP  STORE  *.*  VMS0322
                                  ^-- store all your files in a BACKUP
                                      file on the MSS
                                      (0322 is the date)

                     MSSBACKUP  LIST  VMS0322  KEEP
                                  ^-- list the contents of the above
                                      BACKUP file on MSS at your
                                      terminal, keeping the .MSSBCK
                                      file for later FETCHes today

                     MSSBACKUP  FETCH  VMS0322  RD*
                                  ^-- fetch the files beginning with
                                      RD (do not replace any existing
                                      versions)

                     MSSB  DELETE  VMS0322
                                  ^-- Delete the BACKUP file from MSS


MSSDELETE Delete several MSS files.

          Examples:  MSSDELETE MYFILE
                                  ^-- same as HFT DELETE "MYFILE"

                     MSSD F1,F2,F3,F4,F5
                                  ^-- delete 5 MSS files
```

MSSNEWCHRG

Change the account number on your MSS files.

```
Examples:  MSSNEWCHRG 1222233344 1234567890
                                ^-- change job order number for
                                    all files currently stored with
                                    account number 1-2222-333-44 to
                                    1-2345-678-90
```

*** Using the MSS from the CDC CYBER 860A ***

The MSS is just a peripheral on the CDC CYBER 860A. All files on the CYBER 860A, whether they reside on disk or the MSS, are accessed by the standard NOS permanent file commands.

***** DEC VAXcluster -- VMS *****

The Digital Equipment Corporation (DEC) VAXcluster has four central processing units (CPUs) or nodes which share files and are linked together. There are two VAX 11/780 CPUs, each with 16 megabytes of memory, and two VAX 8550s, each with 48 megabytes of memory. Access is via the DECserver or the DECnet network. A separate VAX 8250 is located in Annapolis and is accessible via the TOFACS selection menu or the DECnet network.

*** VMS Version 4.6 ***

The operating system for the DEC VAXcluster and the VAX 8250 at DTRC is called VMS, version 4.6.

Permanent files (user programs and data files retained for frequent use) reside on disk drives and the Mass Storage System. User files, if not specifically requested on a tape, will be assigned to available disk areas.

*** Accessing the VAXcluster ***

To access the VAXcluster, set your terminal to 8-bit, no parity, then:

- dial (202) 227-5600 <-- this will connect you with the DECserver
- press the RETURN key until it displays a greeting (usually 1-3 times)
- in response to the Username: prompt, enter your User Initials (the DECserver prompt is Local>)
- enter SHOW SERVICES for a list of available services -- as of the publication date, these are:

. 780	- any DTRC VAX 11/780
. 8550	- any DTRC VAX 8550
. CRAY	- any DTRC VAX front-end to the Cray X/MP
. DDN	- Defense Data Network (DT1)
. DT1 or DT2	- DTRC VAX 11/780 DT1 or DT2
. DT3 or DT4	- DTRC VAX 8550 DT3 or DT4
. VAX	- any DTRC VAXcluster node

- enter "CONNECT service" (or "C service") to connect to the desired node
- in response to the Username: prompt, enter your User Initials
- in response to the Password: prompt, enter your login password (the default VAXcluster prompt is $)

*** Login Password ***

Your initial login password is your username, usually your user initials. This is entered in response to

```
Password:
```

the first time you log in. This password MUST be changed during your first session.

To change your login password, type

```
SET PASSWORD
```

You will be prompted for the current password, the new password, and the new password again (to insure there were no transmission problems).

Your password should be changed frequently, and must be changed at least every 90 days.

*** Logout Procedures ***

To terminate your session, get rid of any unwanted permanent files (remember that new versions of a file may be made frequently during the session with up to five retained and costing you money. You may also want to get rid of any journal files (.JOU) made by EDT.

When this is done, or immediately, if the Central Site operator requests it, type LOGOUT. A time and usage summary of the session and a cost estimate will be displayed.

You will be returned to your DECserver session. To leave, type LOGOUT.

Note

> If you do not type anything for about 13 minutes, you will be logged off automatically. You are given a 5-minute warning.

*** System News ***

At login, a system bulletin may be displayed. For more details, type NEWS. To see earlier news items, type OLDNews. To see ancient news items, type VERYoldnews.

*** Login Procedure File ***

A Login Procedure File is a file in your home directory with the name LOGIN.COM. It contains commands to be executed each time you log in before you are given the $ prompt. Commands and qualifiers should be spelled in full to allow for possible future changes in the operating system.

Any command may be in LOGIN.COM. You may want to see who is logged in ($ SHOW USERS), or look at your home directory files ($ DIRECTORY) or all your files ($ DIRECTORY [...]), or define one or more of your HELP libraries ($ ASSIGN UOn:[myid]mylib HLP$LIBRARY_5). You should also define your home directory with a logical name (such as your first name, but NOT your username) using ($ DEFINE myname UOn:[xxxx]). Then, you need only type myname: to refer to your home directory, which you may need to do frequently. For suggestions of other commands, symbols and logical names you might include, type "HELP LOGIN.COM_Hints".

*** Files ***

1. Because VMS automatically deletes the low version number when more than 5 versions of a file are created, you should not use different versions of a file for different purposes. Instead use the file type field.

2. To reduce your file space and, therefore, your costs, you may wish to do a "PURGE [xxxx...]" every now and then to remove all low versions (or "PURGE [xxxx...] /KEEP=2" to keep the highest two versions.

3. When editing with EDT or EVE, a journal file is created of all your editing commands for use in re-editing your file if your editing session is aborted (^Y or a line disconnect). (If your editing session ends normally (EXIT or QUIT), the journal file is deleted.) You should check periodically for any journal files and delete them if they are no longer needed. Use the command "DIRectory /DATE [...]*.JOU,*.TJL" to see them.

*** Batch Jobs ***

A batch job is a procedure which is submitted by the SUBMIT command. By default, the job will be executed on either DT1 or DT2. If your job must run on a specific node, use the /QUEUE=DTn_BATCH qualifier (n is the desired node number). See page 1-3-1 ff for a table of the nodes on which specific software is available.

*** Accessing Other Networks ***

DTRC also has access to the following networks:

```
DDN    - the Defense Data Network (also called INTERNET)
         (host tables allow transfer to some other networks)

TOFACS - the DTRC Office Automation System
```

The following can be reached from our DECnet using SET HOST:

NAVAIR node names: HORNET

NAVSEA node names: SEAHUB, SEAA, SEAB, SEAC, SEAD, SEAE

** Transferring VMS Files To and From TOFACS **

While logged into DT1:

```
ftp dtrc.arpa          <-- File Transfer Protocol to TOFACS
                           (dtrc) via DDN
   -or-
ftp dtrc               <-- via Ethernet
login                  <-- to log into TOFACS
<user name>            <-- your TOFACS user name
<password>             <-- your TOFACS password

get                    <-- get a file from TOFACS
<TOFACS filename>
<VMS filename>

put                    <-- send a file to TOFACS
<VMS filename>
<TOFACS filename>

bye                    <-- leave ftp
```

** Mail to Users at Other Sites **

Mail may be sent to users at other sites which are accessible via DDN. This is one way to transfer large (or small) files.

While logged into VMS:

```
$ mail
MAIL> send
To: wins%"<user@hostname>"   <-- where some typical hostnames
...                              are: dtrc.arpa, tofacsa.arpa,
                                 icst-is.arpa, gwuvax.gwu.edu)
```

For example, to send a message to "sommer" on dtrc (TOFACS B system) from node DT4:

```
$ mail
MAIL> send
To: dt1::wins%"sommer@dtrc"     <-- the brackets are optional
...
```

Mail is sent via the VMS mail utility and the Simple Mail Transfer Protocol (SMTP). The "To:" address has one of the following forms:

Destination	Address Syntax	Utility
same VAX	user	local VMS mail
same network	node::user	DECnet
another VAX	wins%"<user@host>"	SMTP
remote host	wins%"<user@host>"	SMTP
remote host routed through other hosts on your network	wins%"<@host,@host:user@host>"	SMTP
remote host on another network routed through a gateway	wins%"<@host,@gateway:user@host>"	SMTP

SMTP is on node DT1. Therefore, WINS% must be preceded by "DT1::" if used from another node on the VAXcluster. For example, DT1::WINS%"<user@host>".

Note that local VMS and DECnet mail is sent immediately; SMTP mail is sent every 20 minutes.

***** Help Libraries *****

A help library (file type .HLB) contains help modules, that is, modules that provide information about a program, subprogram, procedure, or some general help information such as hints on how to do something. It is created and accessed using the following DCL commands:

LIBRARY	Create, maintain, list, and extract modules from a help library.
HELP	Display the desired helps.

*** The System Help Library ***

The system help library is read using the DCL command HELP. It provides help about the HELP program and lists many topics (VMS features, DCL commands, Hints, and other general information).

*** DTRC Help Libraries ***

Five help libraries have been added to VMS at DTRC:

CCF	- General information about the Computer Center
COS	- Cray COS JCL commands
CRAY	- Routines added to Cray at DTRC
DTLIB	- Subprograms in library DTLIB (Cray, CDC NOS, VMS)
UTILITIES	- Utility programs and procedures

When executing the HELP command, the additional help libraries are accessed by entering '@name', where 'name' is one of the help libraries listed above (e.g., @DTLIB) in response to 'Topic?'. For a table of contents of any of the above libraries, type

```
HELP  @name  Contents
```

*** User Help Module ***

A help module (default file type .HLP) is a file containing all the help information for one or more programs, procedures, etc. Column 1 of each line identifies the different sections of the help module. A digit indicates a keyword; a slash (/) indicates a qualifier; anything else is part of the help text. For example,

```
1 key-1                      <-- HELP topic
    ...
 help message text
    ...
2 key-2                      <-- HELP sub-topic
    ...
 help message text
    ...
n key-n
    ...
 help message text
    ...
1 key-1                      <-- next HELP topic
```

A "1" line gives the topic name (up to 15 characters, avoid using blanks; replace blanks with an underscore (_)). A "2" line is a sub-topic of the "1"-level topic; a "3" line is a sub-topic of the most recent "2"-level sub-topic; etc. Qualifiers (/ in column 1) will be listed separately by HELP and will all be displayed if the (sub)topic they qualify is selected.

A help module might look something like:

```
1 topic
 <description of topic>
2 Qualifiers
 <optional description of qualifiers>
/topic_qualifier_1
 <description of topic_qualifier_1>
/topic_qualifier_2
 <description of topic_qualifier_2>
/topic_qualifier_3
 <description of topic_qualifier_3>
2 sub-topic_1
 <description of sub-topic>
3 sub-topic_of_sub-topic_1
 <description of sub-sub-topic>
3 Qualifiers_of_sub-topic_1
 <optional description of qualifiers>
/sub-topic_1_qualifier_1
 <description of qualifier_1 of sub_topic 1>
/sub-topic_1_qualifier_2
 <description of qualifier_2 of sub_topic 1>
...
```

*** Hints For Designing Help Displays ***

While help messages can continue without interruption, you may wish to format the messages to fit the screen display. A topic ("1" in column 1) will have 17 lines in the first display; a sub-topic ("2" in column 1) will have 15 lines; a sub-sub-topic ("3" in column 1) will have 13 lines; etc. For all levels, the second and following displays have 20 lines. Level 1 lines should not exceed 78 columns; level 2 lines should not exceed 76 columns; level 3 lines, 74 columns; etc. Longer lines may "wrap around".

Every help library should have a module called "HELP" to describe the help library.

You may wish to have a table of contents module (suggested name "Contents") to list the routine names and give a short description of what each routine does.

If possible, the first help screen for a program, subprogram or procedure should contain all that is needed to use it. Definitions of parameters and qualifiers should be put into sub-topics.

*** Selecting (Sub)topic Names ***

While you may choose anything you want for topic and sub-topic names, we recommend the following conventions:

- use upper case for routine names, parameters, and qualifiers (e.g., AUXPRINT, /CC, /HEADER, JGDATE, FLR below)

- use lower case (first letter upper case) for general information (e.g., Parameters, Qualifiers, Examples, Admin_info below)

- replace blanks with underscores (_) so that the name will be listed as a single element by HELP (e.g., Admin_info below)

*** Create a Help Library ***

The LIBRARY command is used to create a help library.

```
LIBRARY /HELP /CREATE  help_library_name

          -or-

LIBRARY /HELP /CREATE=(option,...)  help_library_name
```

where help_library_name is the name of the library to be created. It will have the default filename help_library_name.HLB.

The following options may be specified:

BLOCKS:n The number of 512-word blocks to be allocated.
(default: 100)

HISTORY:n The maximum number of library update history records to be maintained.
(default: 20)

KEYSIZE:n The maximum length of module names.
(default: 15)

MODULES:n The maximum number of modules the library can hold.
(default: 256)

*** Modify a Help Library ***

The LIBRARY command is used to insert, and delete help library modules. Wildcards are allowed in module names.

```
LIBRARY /HELP /INSERT   help_library_name  help_module_name

LIBRARY /HELP /REPLACE  help_library_name  help_module_name

LIBRARY /HELP /DELETE=(module[,...])  help_library_name
```

'LIBRARY /HELP help_library_name help_module_name' is the same as if '/REPLACE' were specified. If '/LOG' is specified, a messages will be displayed for each operation done. (E.g., LIBR /HELP /LOG ...)

*** Compress a Help Library ***

After several inserts, deletes or replaces, there may be a lot of "dead space" in the library. To remove this, that is, to compress the library, use:

```
LIBRARY /HELP /COMPRESS  help_library_name

        -or-

LIBRARY /HELP /COMPRESS=(option,...)  help_library_name
```

/LOG will list the modules as they are copied into the compressed library.

The options available are the same as for /CREATE.

*** List the Contents of a Help Library ***

The LIBRARY command also lists the contents of a help library. The /LIST qualifier, which may be specified alone or with any of the above operations, will provide information about the library including a list of the modules in the library. If /FULL is also specified, the list of modules will include the date and time it was inserted into the library. If /HISTORY is specified, it will show who did what to the library and when. The number of history records retained is defined when the library was created or compressed.

For a list of the library without other operations, use

```
    LIBRARY /HELP /LIST                      -or-
    LIBRARY /HELP /LIST /FULL                -or-
    LIBRARY /HELP /LIST /FULL /HISTORY
```

The list will be displayed on SYS$OUTPUT. To put the listing into a file, use /LIST=filespec.

To list information about specific modules, use /MODULE=(list) where <list> is a comma-separated list of module names with wildcards allowed. The default is /MODULE=*.

To list information about modules inserted after a certain time, use /SINCE (for those inserted today) or /SINCE=date_and_time (for those inserted after a specific date and/or time (e.g., /SINCE=09:00 for those after 9 AM today).

*** Extract a Help Module ***

To extract a help module to make some modifications to it, use

```
    LIBRARY /HELP /EXTRACT=(module[,...])
                  /OUTPUT=file-spec
                  help_library_name
```

If /OUTPUT is specified, the modules are put into file <file-spec>. If /OUTPUT is omitted, they are put into file help_library_name.HLP.

Wildcards are allowed in module names.

*** Accessing your Help Library ***

To access you help library, use

```
    HELP /LIBRARY=filespec  [ topic  [ sub-topic ]  ]
```

where <filespec> must be complete (e.g., U09:[abcd]mylib), not just the filename.

*** Adding Your Help Library to the System Helps ***

The DCL HELP command supports many user libraries in addition to the system library. User libraries are added by assigning help library names to HLP$LIBRARY_n, where n is omitted or a digit. HLP$LIBRARY through HLP$LIBRARY_4 are already defined at LOGIN. You may add your own help libraries starting with HLP$LIBRARY_5. For example, you may wish to put

```
$ DEFINE  /NOLOG  HLP$LIBRARY_5  UOn:[myid]mylib1
$ DEFINE  /NOLOG  HLP$LIBRARY_6  UOn:[myid]mylib2
```

into your LOGIN.COM file so that your help library will always be part of the system HELP command for you. The first missing number (in this case "7") will end the list. These will be listed at the end of the last screen of the topic display. To access library "5" above, use "HELP $mylib1", or "@mylib1" at the Topic? prompt.

*** Using HELP ***

The HELP command access the system help library ("HELP"), your library set ("HELP @libname"), or any other help library ("HELP /LIBRary= filespec").

On initial entry into a help library, the help module is displayed, if present, a list of topics, and, perhaps, the library set. At the "Topic?" prompt, enter the name of the topic for which you want help. Only as many characters as are needed to uniquely identify the topic are required. If the name is not unique, all matching topics are displayed.

After the topic has been displayed (may be more than one screen), a list of additional information (sub-topics) may also be shown. At the prompt, enter the sub-topic name.

When you have finished with a level, press RETURN to go up one level. Pressing RETURN at the "Topic?" prompt exits the HELP command. At any prompt (even in the middle of typing an entry, ^Z (CTRL-Z) will terminate HELP.

Enter a question mark (?) at any time to display the most recent (sub)topic again. The actual help displayed depends on how you got to the current level. The RETURN key should not be pressed with the "?", since the "?" is recognized immediately. (If a help library is entered from a program other than the HELP command, the RETURN is required after the "?".)

If you have forgotten the names of the additional (sub)topics, just enter something you know is not a (sub)topic name (in most cases, "ZZ" is sufficient). This will display an error message and show the valid (sub)topic names.

The up-arrow key ma~ be used to bring back your most-recent entry, which may be edited and resubmitted.

*** Sample Help Modules ***

The following are sample help modules for a program, a subprogram, a procedure, general information; and a HELP help module.

** A Program **

The following is a portion of the help module for the AUXPRINT program.

```
1 AUXPRINT                                               <-- topic
 List a file on an auxiliary printer (one attached to an
 interactive terminal).

   Format:                                   !  Defaults

      AUXPRINT  file-spec  [ /[NO]CC      ]  !  /NOCC
                           [ /[NO]HEADER  ]  !  /NOHEADER
                           [ /LENGTH=1    ]  !  /LENGTH=66
                           [ /SKIP=s      ]  !  /SKIP=0;
                                             !  /SKIP ==> /SKIP=10
                           [ /WIDTH=w     ]  !  /WIDTH=80;
                                             !  /WIDTH ==>
                                                     /WIDTH=132


2 Parameter                                          <-- sub-topic
 file-spec

 Specifies the name of the file to be printed.

 If omitted, you will be prompted for it.

 Defaults:  extender - .DAT; filename - FOR002

2 Qualifiers                                         <-- sub-topic
 The qualifiers may follow the command name or the file-spec.  If a
 qualifier is specified more than once, only the final value is
 retained.

/CC

 /CC
 /NOCC

   Specifies whether the file has carriage control in column 1 of
   each line.

 Default: /NOCC (that is, the file does not have carriage control
          in  column 1)
```

```
/HEADER

 /HEADER
 /NOHEADER

   Determines whether the listing will have a heading giving the
   date and file-spec.

 Default: /NOHEADER

2 Admin_info                                             <-- sub-topic

  Language:     VAX/VMS Fortran 77

  Authors:      Dan Allen       - DTRC Code 189.2
                David V. Sommer - DTRC Code 1893.1

  Date written: 10/81 (da)

  Dates revised
    03/14/85 - dvs     - add qualifiers /CC /HEADER /LENGTH /SKIP
    10/22/85 - dvs     - shorten /CC output by 1 line
               systems - change default to /NOHEADER
    03/07/86 - dvs     - add /WIDTH qualifier
                       - fix /CC processing when first top-of-page
                         is not first record
```

** A Subprogram **

This illustrates a subprogram help module. We suggest that such a help have the following sub-topics:

- . Parameters (if the routine has them)
- . Examples (at least one example to show how to use the routine)
- . Admin_info (to show the source language, author, a brief history, and anything else that might be appropriate)

```
1 JGDATE                                                   <-- topic
 Convert any Gregorian date to a relative Julian number or vice versa.

 Usage:   INTEGER jg, jd, gyear, gmonth, gday
              ...
          CALL JGDATE (jg, jd, gyear, gmonth, gday)

 The relative Julian number corresponding to a Gregorian date is the
 number of days since 11/24/-4713 (extrapolating the Gregorian calendar).

 This subroutine is useful in determining the elapsed number of days
 between any two calendar dates.  It can also be used to find the
 calendar date so many days from any given date.
2 Parameters                                           <-- sub-topic
 CALL JGDATE (jg, jd, gyear, gmonth, gday)

        jg     - in  - int - direction of conversion
                                1 - Gregorian to Relative Julian
                                2 - Relative Julian to Gregorian
  jg=1: jd     - out - int - will contain relative Julian number
        gyear  - in  - int - Gregorian year (e.g., 1985)
        gmonth - in  - int - Gregorian month (1-12)
        gday   - in  - int - Gregorian day (1-31)

  jg=2: jd     - in  - int - relative Julian number
        gyear  - out - int - will contain Gregorian year (e.g., 1985)
        gmonth - out - int - will contain Gregorian month (1-12)
        gday   - out - int - will contain Gregorian day (1-31)

2 Examples                                             <-- sub-topic

        INTEGER jd, gy, gm, gd
           ...
        CALL JDDATE (1, jd, 1985, 2, 25)
        jd = jd + 1000
        CALL JGDATE (2, jd, gy, gm ,gd)

 This example will find the date 1000 days from 02/25/85.
```

```
2 Admin_info                                              <-- sub-topic

Language:     Fortran 77

Author:       David V. Sommer - DTRC Code 1893.1

Date written: 1968 or earlier

Dates revised
   03/01/79 - implement on Burroughs 7700
   02/01/85 - implement on DEC VAXcluster

                        **   A Command Procedure   **

    The procedure FLR has the following definition for all users:

        $ FLR :== @VSYS:FLR

Without this definition, the "Format" would have

        @VSYS:FLR  [ filename]

1 FLR                                                         <-- topic
 Compile Fortran, Link and Run.

    Format:

       FLR  [ filename ]

    If filename is omitted, you will be prompted for it.


 For execution, FOR005, FOR006 and SYS$INPUT are assigned to the
 terminal.  Thus, all Fortran READ, PRINT, READ (5,..., WRITE (6,...,
 TYPE, and ACCEPT statements will read from or write to the terminal.

 Ignore the system message "previous value of SYS$INPUT has been
 superseded".
```

** General Information **

The following is a portion of the help module for a discussion of the DTRC accounting for users with more than one account. This module has no sub-topics.

1 Many_accounts
VAXcluster users with more than one account are assigned a username/password for each account. These usernames differ in the fifth character position, e.g., CAWE, CAWEA, CAWEB. The default login directory for each user is device:[username] where all files owned by the same individual are stored on the same device. For example,

```
     U01:[CAWE]
     U01:[CAWEA]
     U01:[CAWEB]
```

ACCESSING FILES OWNED BY YOUR ALTER EGO

The "usernames" belonging to a particular user are members of a VMS "group". By default on the VAXcluster, members of a group have Read and Execute access to all files owned by their fellow group members. User CAWEA wishing to access a file owned by CAWE simply references [CAWE]file.ext .

Of course, these access rights can be changed by the SET PROTECTION and SET FILE /ACL commands. In addition, all members of these special "groups" have GRPPRV privilege which, when invoked, gives a member of the group full control, including file creation and deletion, over all files owned by all members of the group. GRPPRV is invoked by

```
     $ SET PROCess /PRIVileges=GRPPRV
```

(this would likely be in your LOGIN.COM)

Then to "copy" a file from one account to another, for example CAWE to CAWEA, user CAWEA would

```
     $ COPY  [CAWE]file.ext  []
```

or user CAWE would

```
     $ COPY  file.ext  [CAWEA]
```

To simply "move" a file from one account to another, CAWEA would

```
     $ RENAME  [CAWE]file.ext  []
     $ SET  FILE  /OWNer_uic=CAWEA
```

Finally, the command MYACcount will indicate the account number of the current session or job, while MYACcount /ALL will provide a list of all user/account pairs in the group.

** "HELP" module **

It is recommended, though not necessary, that your help library have a help module named HELP. Such a module will be displayed when you enter the library, and, therefore, should give a brief description of the library and, if appropriate, pointers to related libraries.

The following is the help module HELP for library @CCF:

```
1 HELP
 The CCF help modules provide information of general interest to users
 of the DTRC Central Computing Facility.

 Other help libraries available include:

      @COS        - Cray Operating System JCL
      @CRAY       - DTRC additions to Cray
      @DTLIB      - subprograms in library DTLIB (formerly NSRDC)
      @UTILITIES  - utility programs and procedures

 Last modified: 29-JUN-1988 11:28:21
```

***** Procedures *****

A procedure is a group of control statements in a file (default file type .COM). Calling a procedure provides a simplified way to process that group of control statements. A procedure may call another procedure.

Eight parameters, P1 through P8, are available for you (or another procedure) to pass data or other information to a procedure.

Both string and integer variables may be used in a procedure. Several lexical functions are available to interrogate the system, to manipulate variables, etc. Files may be read or written. And, of course, DCL statements may be executed.

For more information, see AA-Y501A-TE, "Guide to Using DCL and Command Procedures on VAX/VMS".

***** Object Libraries *****

An object library (file type .OLB) contains compiled subprograms for use in linking with a program.

The Librarian utility LIBRARY is used to create, maintain, list, and extract modules from an object library.

*** DTRC Object Library ***

One object library has been added to VMS at DTRC:

```
    VSYS:DTLIB - Subprograms written or maintained by the Computer
                 Center

                 To use: LINK yourobj,DTIB/LIB
```

*** User Object Module ***

An object module (file type .OBJ) is a file containing one or more compiled subprogram(s). They are produced by compiler such as FORTRAN, COBOL, PASCAL, etc.

*** Create an Object Library ***

The LIBRARY command is used to create an object library.

```
    LIBRARY /CREATE  object_library_name

            -or-

    LIBRARY /CREATE=(option,...)  object_library_name
```

where object_library_name is the name of the library to be created. It will have the default filename object_library_name.OLB.

The following options may be specified:

```
    BLOCKS:n   The number of 512-word blocks to be allocated.
               (default: 100)

    GLOBALS:n  The maximum number of global symbols the library can
               contain.
               (default: 128)

    HISTORY:n  The maximum number of library update history records
               to be maintained.
               (default: 20)
```

KEYSIZE:n The maximum length of module names.
(default: 15)

MODULES:n The maximum number of modules the library can hold.
(default: 256)

*** Modify an Object Library ***

The LIBRARY command is used to insert, and delete object library modules. Wildcards are allowed in module names.

```
LIBRARY /INSERT  object_library_name  object_module_file

LIBRARY /REPLACE  object_library_name  object_module_file

LIBRARY /DELETE=(module[,...])  object_library_name
```

'LIBRARY object_library_name object_module_file' is the same as if '/REPLACE' were specified. If '/LOG' is specified, a message will be displayed for each operation. (E.g., LIBR /LOG ...)

If object_module_file contains several object modules, each will be a separate entity in the object library.

If the qualifier /NOGLOBALS is specified, the global symbols for the modules being inserted will not be put into the global symbol table.

*** Compress an Object Library ***

After several inserts, deletes or replaces, there may be a lot of "dead space" in the library. To remove this, that is, to compress the library, use:

```
LIBRARY /COMPRESS  object_library_name

        -or-

LIBRARY /COMPRESS=(option,...)  object_library_name
```

/LOG will list the modules as they are copied into the compressed library.

In addition to the options available for /CREATE:

KEEP Copy the history records, etc., to the compressed library.
(default: do not copy)

*** List the Contents of an Object Library ***

The LIBRARY command also lists the contents of an object library. The /LIST qualifier, which may be specified alone or with any of the above operations, will provide information about the library including a list of the modules in the library. If /FULL is also specified, the list of modules will include the date and timeit was inserted into the library. If /HISTORY is specified, it will show who did what to the library and when. The number of history records retained is defined when the library was created or compressed.

For a list of the library without other operations, use

```
    LIBRARY /LIST                   -or-
    LIBRARY /LIST /FULL             -or-
    LIBRARY /LIST /FULL /HISTORY
```

The list will be displayed on SYS$OUTPUT. To put the listing into a file, use /LIST=file-spec.

If the qualifier /NAMES is specified, the names of all global symbols will also be listed.

*** Extract an Object Module ***

To extract an object module to make some modifications to it, use

```
    LIBRARY /EXTRACT=(module[,...]  /OUTPUT=file-spec
            object_library_name
```

If /OUTPUT is specified, the modules are put into file <file-spec>. If /OUTPUT is omitted, they are put into file object_module_name.OBJ.

*** Linking with an Object Library ***

If your program uses subprograms in an object library, they can be linked using

```
    LINK  your_obj, your_lib/LIBrary
```

where your_obj is the object module for your program
      your_lib is your object library
      /LIBrary tells the linker that your_lib is an object library

If you are linking more than one object file or using more than one object library, you might use one of the following forms:

```
    LINK obj1, obj2, lib1/LIB
    LINK obj1, obj2, lib1/LIB, lib2/LIB
    LINK obj1, obj2, lib1/LIB, obj3
    LINK obj1, obj2, lib1/LIB, obj3, lib3/LIB
```

etc.

***** Text Libraries *****

A text library (file type .TLB) contains text modules, that is, modules containing source programs, documents, notes, data, etc.

The Librarian utility LIBRARY is used to create, maintain, list, and extract modules from a text library.

*** DTRC Text Libraries ***

The following text libraries have been added to VMS at DTRC:

DTLIB - Source code for subprograms in library VSYS:DTLIB.OLB

DTLIBCRAY - Source code for subprograms in library DTLIB on Cray

INCLUDE - Some common block and code segments to INCLUDE in a program or subprogram

UTILITIES - Source code for programs which have been added to VSYS:

*** User Text Module ***

A text module (default file type .TXT) is a file containing a source program, a document, some miscellaneous information, etc.

*** Create a Text Library ***

The LIBRARY command is used to create a text library.

```
    LIBRARY /TEXT /CREATE  text_library_name

            -or-

    LIBRARY /TEXT /CREATE=(option,...)  text_library_name
```

where text_library_name is the name of the library to be created. It will have the default filename text_library_name.TLB.

The following options may be specified:

BLOCKS:n The number of 512-word blocks to be allocated.
(default: 100)

HISTORY:n The maximum number of library update history records to be maintained.
(default: 20)

KEYSIZE:n The maximum length of module names.
(default: 15)

MODULES:n The maximum number of modules the library can hold.
(default: 256)

*** Modify a Text Library ***

The LIBRARY command is used to insert, and delete text library modules.

```
LIBRARY /TEXT text_library_name  text_module_file  /INSERT

LIBRARY /TEXT text_library_name  text_module_file  /INSERT
                                                /MODULE=module_name

LIBRARY /TEXT text_library_name  text_module_file  /REPLACE

LIBRARY /TEXT text_library_name  text_module_file  /REPLACE
                                                /MODULE=module_name

LIBRARY /TEXT text_library_file  /DELETE=(module[,...])
```

"LIBRARY /TEXT text_library_name text_module_file" is the same as if "/REPLACE" were specified. If "/MODULE=..." is omitted, the module name will be the filename without the file type. If "/LOG" is specified, a message will be displayed for each operation. (E.g., LIBR /TEXT /LOG ...)

Wildcards are allowed in the module names when deleting.

*** Compress a Text Library ***

After several inserts, deletes or replaces, there may be a lot of "dead space" in the library. To remove this, that is, to compress the library, use:

```
        LIBRARY /TEXT /COMPRESS  text_library_name

                -or-

        LIBRARY /TEXT /COMPRESS=(option,...)  text_library_name
```

/LOG will list the modules as they are copied into the compressed library.

The options available are the same as for /CREATE.

*** List the Contents of a Text Library ***

The LIBRARY command also lists the contents of a text library. The /LIST qualifier, which may be specified alone or with any of the above operations, will provide information about the library including a list of the modules in the library. If /FULL is also specified, the list of modules will include the date and time it was inserted into the library. If /HISTORY is specified, it will show who did what to the library and when. The number of history records retained is defined when the library was created or compressed.

For a list of the library without other operations, use

```
        LIBRARY /TEXT /LIST                     -or-
        LIBRARY /TEXT /LIST /FULL               -or-
        LIBRARY /TEXT /LIST /FULL /HISTORY
```

The list will be displayed on SYS$OUTPUT. To put the listing into a file, use /LIST=file-spec.

*** Extract a Text Module ***

To extract a text module to make some modifications to it, use

```
        LIBRARY /TEXT /EXTRACT=(module[,...]) /OUTPUT=file-spec
                text_library_name
```

If /OUTPUT is specified, the modules are put into file <file-spec>. If /OUTPUT is omitted, they are put into file text_library_name.TXT.

Wildcards are allowed in the module names.

***** CDC CYBER 180/860A -- NOS *****

The Control Data Corporation (CDC) CYBER 180 model 860A has a single central processing unit (CPU) and 16000000 octal (2,097,120) 60-bit words of memory, of which 400000 octal is addressable by each job.

The CPU has 24 registers for operating on information: 8 address (A), 8 operand (X) and 8 increment (B) registers. The CYBER 860A has a buffer of 12 central memory (CM) words of instructions, called an instruction stack, and a 2048-word, high-speed cache memory.

Peripheral processors (PPs) are small computers (4096 12-bit words of memory) which handle most input and output (I/O). There are 20 normal and 5 concurrent PPs on the CYBER 860A.

There are 28 normal and 5 concurrent I/O channels. Most peripheral equipment interfaces with the central system through the PPs via the I/O channels. The printers and remote terminals interface with the system via CDCnet.

*** NOS Version 2.5.3 ***

The operating system for the CDC CYBER 860A at DTRC is called the Network Operating System, version 2.5.3 (NOS 2.5 - level 688) and differs only slightly from the standard NOS system. The interactive subsystem for teletype-compatible terminals is called IAF (InterActive Facility); the subsystem for medium-speed remote batch terminals is called RBF (Remote Batch Facility).

Permanent files (user programs and data files retained for frequent use) reside on model 895 disk drives and the Mass Storage System. User files, if not specifically requested on a tape, will be assigned to available disk areas.

*** Accessing the CDC 860A ***

To access the CDC CYBER 860A:

- dial (202) 227-4800 <-- this will connect you with CDCnet

- press the RETURN key until it displays a greeting (usually two times) -- (CDCnet has no prompt, but HELP is available)

- enter DO DEC_VT100 (if you are using a DEC VT100-compatible terminal; this will set half duplex (echo on) for your session without having to change your terminal set-up)
  (another way is to put "TRMDEF,EP=Y." into your LOGINPR file -- this is preferred, since you then don't have to bother with the DO command each time you log in)

- enter CREC NOS to connect to the CDC 860A NOS system

- in response to the Family: prompt, enter either
  - ,xxxx,pw,IAF (xxxx is your User Initials,
    pw is your login password
    IAF is the InterActive Facility)

  -or-
  - RETURN, then the rest of the information one item at a time as prompted

- in response to the CHARGE NUMBER: prompt, enter your Job Order Number

- when you receive the "/" prompt, you are in IAF
  - if you entered your Job Order Number incorrectly, you must enter "CHARGE,number." at the "/" prompt until a valid number is accepted.

*** Terminal Keys ***

NOS supports screen formatting for most display terminals. Many commands use a full-screen mode when the SCREEN command is used. When these commands show function keys, they are shown as they appear on a CDC Viking 721 terminal. When using other terminals, different keys or sequence of keys may be needed for the desired function.

The following table shows the key(s) to be used for some terminals at DTRC. The DT100 keypad for use in FSE is on page 5-1-5.

CDC Viking 721	DEC VT100	Tektronix T4115	CDC Viking 721	DEC VT100	Tektronix T4115
F1	keypad 1 + RETURN	F1	shift F1	PF1 + RETURN	shift F1
F2	keypad 2 + RETURN	F2	shift F2	PF2 + RETURN	shift F2
F3	keypad 3 + RETURN	F3	shift F3	PF3 + RETURN	shift F3
F4	keypad 4 + RETURN	F4	shift F4	PF4 + RETURN	shift F4
F5	keypad 5 + RETURN	F5	shift F5	keypad - + RETURN	shift F5
F6	keypad 6 + RETURN	F6	shift F6	keypad , + RETURN	shift F6
F7	keypad 7 + RETURN	F7	shift F7	kp ENTER + RETURN	shift F7
F8	keypad 8 + RETURN	F8	shift F8	keypad . + RETURN	shift F8
F9	keypad 9 + RETURN	ctrl A	shift F9		ctrl Q
F10		ctrl S	shift F10		ctrl W
F11		ctrl D	shift F11		ctrl E
F12		ctrl F	shift F12		ctrl R

CDC	DEC	Tektronix	CDC	DEC	Tektronix
Viking 721	VT100	T4115	Viking 721	VT100	T4115
F13			shift F13		
F14			shift F14		
F15			shift F15		
F16			shift F16		
NEXT	RETURN	RETURN			
HELP			shift HELP		
BACK			shift BACK		
STOP ctrl T+NEXT	ctrl T + RETURN	ctrl T + RETURN	shift STOP	ctrl T + RETURN	ctrl T + RETURN
FWD			shift FWD		
BKW			shift BKW		
UP			shift UP		
DOWN			shift DOWN		
			shift CLEAR		

SCREEN,DT100 before entering FSE puts you in full-screen mode with the following definition of the keypad.

```
+---------+---------+---------+---------+
|PF1      |PF2      |PF3      |PF4      |
|         |         |         |         |
|  del b  |  join   |  del c  |  del w  |
+---------+---------+---------+---------+
|7        |8        |9        |-        |
|         |         |         |         |
|  ins b  |  split  |  ins c  |  ins w  |
+---------+---------+---------+---------+
|4        |5        |6        |,        |
|         |         |         |         |
| mark c  |  move   |  del 1  |   pos   |
+---------+---------+---------+---------+
|1        |2        |3        |ENTER    |
|         |         |         |         |
| mark 1  |  copy   |  ins 1  |         |
+---------+---------+---------+         |
|0                  |.        |         |
|                   |         |         |
|     forward       |  back   |  home   |
+-------------------+---------+---------+
```

After pressing one or more of the keypad keys, the RETURN key must be used to perform the requested functions.

The arrow keys may be used to position the cursor.

*** Direct versus Indirect Files ***

Unlike most other operating systems, NOS supports two distinct types of file: direct and indirect.

Disk space is allocated by PRU (physical record unit) with one PRU holding 64 words (640 6-bit or 320 8/12-bit characters).

A direct file (not to be confused with a "direct access" or random file) occupies one or more blocks of 704 PRUs and is charged by the number of blocks needed to hold the file. (A 705-PRU file occupies 2 blocks.) When you ATTACH a direct file, you are working with the actual file. Changes made by a program immediately change the actual file and cannot be "undone". Changes made while editing alter the file with you QUIT FSE. To "undo" the changes before QUITting, enter "SET FILE dummy" (SF dummy) and the changes will be made to local file "dummy".

An indirect file occupies up to 696 PRUs and is charged by the number of PRUs needed to hold the file. (A 1-PRU indirect file occupies 1 PRU, while a 1-PRU direct file occupies 1 block, or 704 PRUs.) When you GET an indirect file, you are working with a copy of the file. Any changes made by editing or by a program affect only the copy and may be "undone" any time prior to REPLACE-ing the file. An indirect file is actually a portion of a larger "file" containing other indirect files. You cannot work with the actual file because a change could lengthen it, thus destroying the file which physically follows it. When you REPLACE an indirect file, the new file is put wherever there is room in the larger "file". Notice that the largest indirect file fits within one block.

Direct files are required for files larger than 696 PRUs and for files which require that changes be made in real time, perhaps for other users of the file.

Indirect files are recommended for short files. They are especially useful for source programs and data files which are under development, where you might want to try some changes but not make them permanent until you decide. The largest indirect file actually holds a lot of information (445,440 6-bit, 222,720 8/12-bit characters). For example, the CDC 750 NOS/BE public procedure file had some 157 procedures, some of them quite elaborate, and occupied 684 PRUs.

***** NOS CCL Commands *****

The NOS CYBER Control Language (CCL) statements are grouped by function in this section. See Appendix D for a description of the syntax for each command. (DTRC) indicates a command or program added at DTRC.

*** Flow Control ***

BEGIN Transfer control to a procedure.

DISPLAY Evaluate an expression and put the result into the job's dayfile in octal and decimal.

ELSE Terminate skipping (false IF command with same label), or initiate skipping (true IF command with same label) to ENDIF with same label.

ENDIF Terminate skipping by a SKIP, IF, or ELSE command with a matching label.

ENDW The end of a WHILE loop.

EXIT Resume processing commands after a previous error.

IF Conditionally skip one or more commands.

name Transfer control to a procedure.

NOEXIT Continue processing with the next commano even if an error has occurred (suppress EXIT processing).

ONEXIT Reverse the effect of NOEXIT.

REVERT Return from a procedure.

SET Assign a value to a control register, an error flag, or the enter-skipped-commands-in-the-dayfile flag.

SKIP Unconditionally skip succeeding commands, ending with an ENDIF with a matching label.

WHILE Start of a command loop.

*** Job Control ***

* Entire line is a comment.

BLOCK Add one or more lines of 10x10 block letters to a file.

CHARGE Validate charging information for the job.

COMMENT Place a comment in the system dayfile and the dayfile for any of your jobs.

CSUBMIT Submit a job to a Cray mainframe.

CTIME Put the accumulated CPU time (in seconds) into the job's dayfile.

DAYFILE Write the job's dayfile to a file.

DROP Drop any of your executing or queued files (except the job issuing the DROP command).

ENQUIRE Get information about your jobs.

ENTER Enter a series of commands on one line.

ERRMSG Control the display of error messages in a procedure.

GO Clear the pause bit of one of your jobs.

job Identifies requirements for a batch job.

LENGTH Gives the current status of one of your local files.

LIMITS List your validation limits.

LISTLID List network configuration and host availability information.

MFL Reset maximum field length for subsequent job steps.

NORERUN Clear the job rerun status.

NOTE Create a file with the command line containing the lines for the new file.

OFFSW Clear sense switches.

ONSW Set sense switches.

PASSWOR Change your password.

PAUSE Set the pause bit of one of your executing jobs.

QGET Assign a queued file to your job.

RERUN Allow a job to be rerun if necessary.

RESOURC Specify that more than one tape drive is required.

RFL Set running field length.

RTIME Put the real-time clock time into the dayfile.

SETASL Set the SRU limit for an accounting block.

SETCORE	Preset each word of the field length except for RA+2.
SETJOB	Change some of the current job's attributes.
SETJSL	Set the SRU limit for each subsequent job step.
SETPR	Decrease the CPU priority of a job.
SETTL	Set the CPU time limit for each subsequent job step.
STIME	Put the accumulated SRU value for the job into the dayfile.
SUBMIT	Put a job into the input queue.
SWITCH	Set sense switches.
UPROC	Specify a user prologue to be executed each time you start a job.
USER	Identify you and provide validation information for each batch job.

*** Interactive ***

** Terminal Control **

ASCII	Set terminal to ASCII.
CSET	Change the terminal's character set mode.
LINE	Set your terminal for line (or scrolling) mode for FSE and HELPME.
NORMAL	Reverse the effect of ASCII, AUTO, BRIEF, and CSET,ASCII commands.
SCREEN	Set your terminal for screen mode.
TDU	Compile a terminal definition file and store it in a user library which can later be accessed by a SCREEN or LINE command.
TRMDEF	Change terminal characteristics.
%1	Interrupt current job step.
%2	Terminate current job step.
%HELP	Display the CDCnet command list.

** Subsystem Selection **

ACCESS	Select the ACCESS subsystem.
BASIC	Select the BASIC subsystem.
BATCH	Select the BATCH subsystem.
EXECUTE	Select the EXECUTE subsystem.
FORTRAN	Select the FORTRAN subsystem.
NULL	Select the NULL subsystem.

** Interactive Status **

%D	Immediately detach a terminal job from the terminal.
%E	Immediate detailed job status.
%S	Immediate abbreviated job status.

** Job Processing **

APPSW	Switch temporarily to an alternate NAM application program.
BYE	Terminate an application.
DIAL	Send a one-line message to another user.
EXPLAIN	Retrieve an on-line version of a CDC manual.
GOODBYE	Terminate an application.
HELLO	Logs you out of IAF and switches you to another application, or starts another login.
HELP	Ask for help.
HELPBE	On-line help for the NOS-equivalent of NOS/BE commands.
HELPME	Display a brief description of a command, prompt for parameters, execute the command.
LIST	List lines of a local file.
LOGIN	Terminate your current application and start another.
LOGOUT	Terminate an application.
RECOVER	Recover a detached job or interrupted terminal session.

REDO Modify and re-execute a previously entered command without having to retype the entire command.

SHOW Display a screen formatting panel for testing purposes.

WHATJSN Get the job sequence number for the specified user name.

X Execute a batch command.

XMODEM Transfer a file between NOS and a PC using the Christensen protocol.

*** File Management ***

ASSIGN Assign a file to a device.

BKSP Backspace a file (by logical records).

CLEAR Release all (or all but one or more specified) auto-drop files assigned to the job.

COPY Copy data from one file to another.

COPYBF Copy a multi-file file.

COPYBR Copy a records from one file to another.

COPYCF Copy a coded multi-file file.

COPYCR Copy a records from one coded file to another.

COPYEI Copy a file through end-of-information.

COPYSBF Copy a file, shifting the lines one character to the right for printing on a printer.

COPYX Copy a file until a user-specified condition is met.

FCOPY Convert a file from one character set to another.

FILE (CRM) Describe the attributes of a file.

LO72 Reformat files.

LOCK Prevent writing on a local file.

OUT Send deferred output files to the print or punch queue immediately.

OVWRITE Overwrite files to destroy their contents.

PACK Remove all EORs and EOFs from a file.

RENAME Change the name of a local file.

REQUEST Assign a file to receive checkpoint dumps, or send a message to the operator to assign to the described device.

RETURN Release files (and file space depending on file type) assigned to a job.

REWIND Position files at beginning-of-information (BOI).

ROUTE Direct the disposition of an indirect file and define its characteristics.

SCOPY Copy coded file(s) displaying EORs and EOFs in the receiving file.

SETFS Set the auto-drop/no-auto-drop status of files assigned to your job.

SKIPEI Position a file at end-of-information.

SKIPF Skip forward a specified number of files.

SKIPFB Skip backward a specified number of files.

SKIPR Skip forward a specified number of record or file marks.

TCOPY Copy X (binary), E, B, or SI files to disk, I, or SI (binary) tape.

TDUMP Octal or alphanumeric dump of all or part of a file.

UNLOAD Release files assigned to your job and perhaps their file space.

UNLOCK Rescind the LOCK command and clear the write interlock for specified local disk files.

VERIFY Binary file comparison.

WRITEF Write a specified number of file marks on a file.

WRITER Write a specified number of empty records on a file.

*** Permanent File ***

APPEND Append information to the end of an indirect access file without retrieving the file.

ASSIGN Assign a file to a device.

ATTACH Assign a direct access permanent file to a job.

CATLIST List permanent file information.

CHANGE Change some characteristics of a permanent file.

DEFINE Create an empty direct access permanent file.

GET Get copies of indirect access permanent files as local files.

PERMIT Explicitly permit another user to access one of your private files.

PURGALL Purge all your files which match the parameters.

PURGE Purge one or more direct or indirect permanent files.

RECLAIM Selectively backup and reload local and permanent files.

REPLACE Purge an indirect access file and replace it with a copy of a local file; save a copy of a local file as a new indirect access file.

SAVE Put a copy of a local file on disk as an indirect access file.

*** Load/Dump Memory ***

DMB Binary dump of exchange package and central memory.

DMD Dump the exchange package or central memory in both octal and display code.

DMP Dump the exchange package or central memory in octal.

*** Tape Management ***

ASSIGN Assign a file to a device.

BLANK Blank label a magnetic tape.

LABEL Mount a magnetic tape and, if labelled, check the label.

LISTLB List labels of an ANSI-labelled tape.

REQUEST Assign a file to receive checkpoint dumps, or send a message to the operator to assign to the described device.

REQUEST Request a tape be mounted (LABEL is preferred).

VSN Associate a local file name with one or more volume serial numbers.

*** Checkpoint/Restart ***

CKP Take a checkpoint dump.

RESTART Restart a checkpointed job.

*** Procedures ***

BEGIN Transfer control to a procedure.

REVERT Return from a procedure.

*** System Utilities ***

FSE Invoke the full screen editor.

UPDATE Create, edit or copy an Update-formatted program library.

*** Library Maintenance ***

CATALOG List information about each record in a file.

COPYL Selective single replacement of object modules.

COPYLM Selective multiple replacement of object modules.

GTR Selective extraction of records from a file.

ITEMIZE List information about each record of a binary file.

LIBEDIT Create and maintain a library of programs, subprograms, procedures, or text.

LIBGEN Create a new user library of routines for use by the loader.

LIBRARY (Loader) Specify a set of global libraries to be searched for externals and programs and the order in which they are to be considered.

ULIB Create a user library; add, delete or replace a record.

VFYLIB List differences in name, type, length, and checksum for the records of two files.

*** Programming Languages ***

COBOL5 Compile COBOL 74 program.

FTN5 Compile Fortran 77 program.

X,BASIC Compile a BASIC program without changing to the BASIC subsystem.

*** Loader and Loader-related Control Statements ***

EXECUTE Complete loading, fill unsatisfied references by system (and user) library search, generate load map and execute the program.

LDSET Set any of several loader options for the current load only.

LGO Load and execute the default compiler binary output file.

LIBLOAD Load modules from specified library which contains the specified entry points.

LIBRARY Specify a set of global libraries to be searched for externals and programs and the order in which the libraries are to be searched.

LOAD A list of files whose contents are to be loaded.

MAP Specify the global default option for load maps.

name Load and execute binary program or procedure in local file <name>.

NOGO Complete the loading of a program, including generating load map, but do not execute.

REDUCE Turn the reduce flag on or off.

RFL Set running field length.

SATISFY Satisfy unsatisfied externals prior to normal satisfaction at load completion.

SLOAD Selectively load modules from local file <lfn>.

***** Procedures *****

A procedure is a group of control statements separate from the job control statement file. Calling a procedure provides a simplified way to process that group of control statements. A procedure may be called by a job repeatedly, by another procedure, or by itself.

In general, the "CCL CYBER Control Language Reference Guide" for NOS/BE can be used for NOS. It is available from User Services. See also NOS 2 Reference Set Volume 3: System for additional features.

*** Procedure Directives ***

Procedure directives allow you to control procedure processing options. The procedure "title", the help text, and all "text" and "message"s may be in 6/12-bit upper and lower case.

.CC(n) Specify the concatenation character for a procedure.

.CORRECT,text.
.CORRECT=text.
Specify the prompt to follow an incorrect procedure parameter entry for an interactive procedure.

.DATA,lfn.
Create a local file from a procedure.

.DATA is terminated by another .DATA, an end-of-record (not .EOR), an end-of-file (not .EOF), or end-of-information.

.IF, .ELSE, .ENDIF can be used within the data lines for conditional inclusion.

.ELSE,label.
End skipping by a matching .IF or start skipping to a matching .ENDIF in a procedure.

.ENDHELP.
Mark the end of the help section of an interactive procedure.

.ENDIF,label.
End skipping from a matching .IF or .ELSE in a procedure.

.ENTER,text.
.ENTER=text.

Specify the prompt for before an interactive procedure parameter entry.

.EOF. Put an end-of-file into a file created by .DATA in a procedure or in the procedure command record.

.EOR. Put an end-of-record into a file created by .DATA in a procedure or in the procedure command record.

.EX.command.

Submit a command to the system for immediate execution.

.EXPAND,option

End or resume procedure expansion.

.Fn,text.
.Fn=text.

Specify a label for one of the six programmable function keys for use with screen mode parameter displays in an interactive or menu procedure.

On a VT-100, these correspond to keypad keys 1-6.

.HELP.
.HELP,NOLIST.
.HELP,parm.
.HELP,parm,NOLIST.

Specify the help (upper and lower case) text for a procedure or parameter.

.IC(n)

Specify the inhibit character for a procedure.

```
.IF,expression.command.    <-- note 2 terminators
.IF,expression,label.      <-- only 1 terminator
```

Conditional expansion of a procedure.

.NOCLR,message.
.NOCLR=message.

Inhibit automatic screen clearing during a procedure.

.NOTE,message.
.NOTE=message.

Specify a message to be displayed on the screen and in your dayfile at the end of a procedure call (when all required parameters are supplied).

Use the NOTE command to display comments during the execution of a procedure.

.PAGE,text.
.PAGE=text.

Specify the string to precede the page number on the screen.

```
.PROC,pname*I"title",p1,p2,...,pn.ck.             <-- interactive
.PROC,pname*M"title",keyword=(selections).ck.     <-- menu
.PROC,pname,p1,p2,...,pn.                         <-- passive
         The procedure header specifying the procedure name and
         parameters, and enabling parameter prompting.

         Parameters:  pname      - the name of the procedure
                                   (1-7 alphanumerics, first should be
                                   alphabetic; append *I for
                                   interactive, *M for menu-driven;
                                   nothing for passive)

                      title      - the procedure title
                                   (default: pname)

                      pi         - up to 50 parameters, each of the
                                   form:
                                     Interactive:
                                       keyword"description"=(checklist)
                                       keyword'description'=(checklist)
                                     Passive:
                                       keyword
                                       keyword=
                                       keyword=default1
                                       keyword=default1/default2
                                       keyword=/default2
                                       keyword=#DATA
                                       keyword=#FILE
                                     keyword     - 1-10 alphanumeric
                                                   characters
                                     description - parameter prompt
                                                   (see title above)
                                     checklist   - a list of acceptable
                                                   values and the
                                                   parameter syntax
                                     default1    - 1-40 chars if pi
                                                   is omitted
                                     default2    - 1-40 chars if pi is
                                                   specified without
                                                   value or with the
                                                   value pi
                                     #DATA       - the name of an
                                                   unnamed .DATA file
                                     #FILE       - the file containing
                                                   pname
                      selections - the menu selections in the form:
                                     n1"desc1",n2"desc2",...,nn"descn"
                                   where ni    - integer (1-10 digits)
                                                 identifying the menu
                                                 selection
                                         desci - the menu item descrip-
                                                 tion (see title above)
                      ck         - comment keyword (1-10 characters)
```

.PROMPT,text.
.PROMPT=text.

Specify the text for the general request for input in a procedure.

.SET,keywd_1=strexp_1,...,keywd_n=strexp_n.

Build new procedure parameters.

.*comment

A comment in a procedure.

```
Example: .PROC,myproc,....
           ...
         REVERT...myproc
         .*
         .* created 88/04/12
         .* last modified 88/05/20 (add "PW" parameter)
         .*
         .* End of myproc
```

*** DTRC Procedure Library ***

Public-access procedure library PROCFIL has been added to NOS at DTRC and will be searched if you do not have a local or permanent file named PROCFIL.

*** Sample Procedure ***

The following illustrates a simple interactive procedure to compile a Fortran program and, optionally execute the program.

```
.PROC,F5*I,I'Input'=(*F,*N=INPUT),
           B'Binaries'=(*F,*N=LGO),
           L'Output'=(*F,*N=OUTPUT),
           LO'List options'=(*n=0,0,O,R,A,M,S),
           GO'Execute'=(*N=).
FTN5,#I=I,#B=B,#L=L,#LO=LO.
IF,$GO$.NE.$$.LGO.
REVERT...F5
```

Invoking this procedure with

```
f5,?
```

causes the following dialog:

```
PARAMETERS FOR F5 ARE  I, B, L, LO, GO
Input? test
Binaries? bfile
Output? <press the RETURN key for the default>
List options? s
Execute? y
```

The generated FTN5 statement will be:

```
FTN5,I=TEST,B=BFILE,L=OUTPUT,LO=S.
```

Since GO is non-null, "LGO." will also be executed.

***** Program Libraries *****

Source programs and data may be in separate datasets or may be stored and maintained in program libraries. UPDATE creates and maintains these libraries, which may be display code or ASCII (8/12).

*** UPDATE ***

UPDATE is a program for creating and modifying a program library (PL). In addition, UPDATE will extract individual modules for input to a compiler or other program.

By default, 72 columns of information are retained. Fifteen additional characters are retained for each line: an 9-character identifier, a 6-digit sequence number, i.e., id_seq, and is often referenced as id.seq.

UPDATE supports two kinds of text modules or decks:

- a regular deck (beginning with a DECK directive)
- a common deck (beginning with a COMDECK directive) which may be included in decks with a CALL directive

Each type includes all lines following the deck directive until the next deck or modification directive.

History information is retained allowing the deletion, modification, or restoration of previous modifications.

See Appendix D for a description of the UPDATE control statement parameters.

*** UPDATE Directives ***

An UPDATE directive has the following format:

```
m directive_name  [ parameters ]
```

where m is the master character (default: asterisk (*)). There are five categories of directives.

** DECK and COMDECK **

*DECK deck (*DK)
First line of a new deck. <deck> is up to 9 characters except comma, period, blank, colon, equals.

*COMDECK cmdk (*CDK)
First line of a new common deck.

** Compile File **

*CALL cmdk (*CA)
Include the contents of a common deck.

*COMPILE p1,p2,...,pj.pk,...,pn (*C)
Write one or more decks, including a range (pj.pk), to the compile and/or source datasets. Use UPDATE,K to force the output order.

*CWEOF (*CW)
Write an EOF on the compile dataset if anything was written since the last EOF.

*WEOF (*W)
Write an EOF on the compile dataset.

*WIDTH linelen,idlen (*WI)
Change the data and id length (default: 72,4).

*DO, *DONT, *IF, and *ENDIF are also available.

** Modification **

*ADDFILE lfn,name (*AF)
Read creation directives and text from file lfn and insert after the specified deck or line.

*BEFORE id.seq (*B)
Insert before a line.

*CHANGE oldid,newid,...,oldid,newid
Change correction set identifier.

*COPY dk,id1.seq1,id2.seq2 (*CY)
Copy a range of lines from deck or comdeck <dk>.

```
*DELETE id1.seq1                (*D)      <-- one line
*DELETE id1.seq1,id2.seq2                 <-- a range of lines
*DELETE id1.seq1,.seq2                    <-- same (short form)
```

Delete a line or a range of lines.

*IDENT ident (*ID)
*IDENT ident,B=num,K=id,U=id
Identify a set of modifications. You can specify a sequence number bias, and require that other modification sets be known (K=) or unknown (U=).

*INSERT id.seq (*I)
Insert after a line.

*MOVE dk1,dk2 (*M)
Move deck <dk1> to follow deck <dk2>.

*PURDECK dk,dk2,...,dkj.dkk,...,dkn (*PD)

```
              Permanently remove decks.

*PURGE id1,id2,...,idj.idk,...,idn     (*P)
              Remove the effect of a modification set (idi), a range of
              datasets (idj.idk), or a set and all following (idn=*).

*RESTORE id1.seq1                (*R)     <-- one line
*RESTORE id1.seq1,id2.seq2                <-- a range of lines
              Restore a line or a range of lines.

*SEQUENCE id1,id2,...,idj.idk,...,idn..     (*S)
              Resequence active lines and purge inactive lines in the
              specified decks.

*YANK id1,id2,...,idj.idk,...,idn
              Temporarily delete a deck, comdeck, or modification set
              previously yanked.

*YANKDECK dk1,dk2,...,dkj.dkk,...,dkn
              Temporarily deactivate decks.

*SELPURGE, and *SELYANK are also available.
```

**   File Manipulation   **

```
*COPY name,id1.seq1,id2.seq2,lfn         (*CY)
              Copy a range of lines from deck or comdeck <name> to file
              <lfn>.

*READ lfn    (*RD)
              Read input from another file.

*REWIND lfn
              Rewind a file.

*SKIPF lfn,n
              Skip record(s) in a local file.
```

**   Input Stream Directives   **

```
*ABBREV
              Resume recognition of abbreviations.

*ENDTEXT     (*ET)
              End a *TEXT section.

*LIST     (*L)
              Resume listing input lines.  UPDATE,L=0 overrides *LIST.

*NOABBREV    (*NA)
              Do not check for abbreviation.

*NOLIST     (*NL)
```

Stop listing input lines.

*TEXT (*T)
Treat all statements between *TEXT and *ENDTEXT as text.

*SKIP and *ENDSKIP are also available.

** Special **

*LIMIT n (*LT)
Limit the output listing to n lines.

*/comment
A comment line.

*DECLARE, *DEFINE, and *PULLMOD are also available.

*** Examples ***

1) Create a PL:

```
jobnam1,....
USER,user,pw.
CHARGE,....
UPDATE,P=0,C=0.                 <-- no OLDPL or COMPILE
SAVE,NEWPL=mypl.                <-- create indirect file
<eor>
*DECK DECK1
lines for deck DECK1
*DK DECK2
lines for deck DECK2
*DK DECK3
lines for deck DECK3
```

2) Interactively extract, compile and execute deck DECK2 from PL MYPL:

```
GET,OLDPL=mypl.                <-- get indirect file
NOTE,uin./*COMPILE deck2
UPDATE,I=uin.
FTN5,I.
LGO.
```

3) Create a PL using a common deck, compile and execute:

```
jobnam3,....
USER,user,pw.
CHARGE,....
PURGE,mypl/NA.
DEFINE,NEWPL=mypl.              <-- direct file
UPDATE,P=0.                     <-- no OLDPL
FTN5,I.
LGO.
<eor>
*COMDECK COM3
      common / mycom / a, b
      real a, b
*DK PROG3
      program prog3
*CALL COM3
      call sub
      print *, 'a,b=', a, b
      end
*DECK SUB
      subroutine sub
*CA COM3
      a = 1.
      b = 2.
      return
      end
<eoi>
```

4) Update old source library to new, compile all decks and execute:

```
jobnam4,....
USER,user,pw.
CHARGE,....
GET,OLDPL=mypl.                 <-- get indirect old library
UPDATE,F,N.
FTN5,I.
LGO.
REPLACE,NEWPL=mypl.             <-- replace indirect old library
<eor>
*IDENT DS0620         <-- correction must be unique (initials,date)
*INSERT ALONE.57      <-- correct deck ALONE by insert after line 57
   (Fortran statements)
*DELETE FOUR.12,13    <-- correct deck FOUR replacing lines 12-13
   (new lines to replace deletions - optional)
<eor>
   (data lines, if any)
<eoi>
```

5) Select routines from source subroutine library and compile with your own program:

```
jobnam5,....
USER,user,pw.
CHARGE,....
FTN5.                   <-- compile your own programs
ATTACH,thatpl/UN=NSYS.
UPDATE,P=thatpl,Q,L=0.
FTN5,I.
LGO.
<eor>
    (own Fortran decks)
<eor>
*C rtnl,rtn6.rtn8      <-- select decks RTN1, 6, 7, 8 from library
<eor>
    (data records, if any)
<eoi>
```

***** Object Libraries *****

LIBEDIT and LIBGEN are utilities for creating and maintaining libraries of absolute and relocatable object modules. These libraries can then be used by the loader to locate the program to execute or the subprograms to be loaded with your program.

See Appendix B for the LIBEDIT and LIBGEN control statements.

*** LIBEDIT Directives ***

The following are used in the descriptions of the LIBEDIT directives:

rid - record identifier

format	meaning
type/name	the record has this type and name
name	the record has this name and the default type
*	end-of-file (*BEFORE only)

gid - group identifier

format	meaning
type/name	the record with this type and name
name	the record with this name and the default type
type1/name1-type2/name2	a group of records
type1/name1-name2	a group of records of type1
name1-name2	a group of records with the default type
type/name-*	all records of this type beginning with <name>
name-*	all records of the default type beginning with <name>
type/*	all records of this type
*	all records
0	insert a zero-length record

The following are some of the LIBEDIT directives. Directives start with an asterisk in column 1, followed by the directive name (or abbreviation). Directives can be continued (gid entries cannot be split). For example:

```
*BEFORE,ov1/p1,ov1/p2
ov1/p3
```

*ADD LIBn,gid1,gid2,...

Append records to a record group.

Parameters: LIBn - a record group (from a CATALOG listing) (1 <= n <= 63)

gidi - records from the current replacement file to be appended

*BEFORE rid,gid1,gid2,... (*B)

Insert records before a specified record.

*BUILD dname

Build a directory at the end of the new file.

Parameters: dname - the name for the directory record (1-7 alphanumerics)

*COMMENT rid comment

Add a comment to the prefix table.

Parameters: comment - up to 70 characters with excess truncated

*COPY Copy the new file to the old file after editing.

Remarks: *COPY is the same as LIBEDIT,...,C.

*DATE rid comment

Add the date and a comment to the prefix table.

Parameters: comment - up to 70 characters with the excess truncated

*DELETE gid1,gid2,... (*D)

Do not copy the specified records to the new file.

*FILE lfn The name of the file containing the replacement records.

Parameters: lfn - use * for the replacement file from the LIBEDIT command (B=)
(default: LGO)

*IGNORE gid1,gid2,...
Ignore specified records in the replacement file.

Examples:

```
*FILE myrecs
*IGNORE D-*
            ^-- ignore all records from D to the
                end-of-file
```

*INSERT rid,gid1,gid2,... (*I, *AFTER, *A)
Place the replacement records after the specified groups in the new file.

*LIBGEN record_name
Generate a user library (using LIBGEN) after processing.

Parameters: record_name - the name of the new user library directory record

Remarks: *LIBGEN overrides *VERIFY.

*LIST list_file,list_opt
Specify the list file and the list options.

Parameters: list_file - same as LIBEDIT,L=

list_opt - same as LIBEDIT,LO=

*NEW newfile
Specify the name of the new file.

Parameters: newfile - same as LIBEDIT,N=

*NOINS Prevent the insertion of unreplaceable records.

Remarks: same as LIBEDIT,NI

*NOREP lfn1,lfn2,...
Do not automatically replace records from the specified files.

Remarks: Records from these files can be copied to the new file only by using *AFTER, *BEFORE, *INSERT, or *REPLACE.

*NOREW Do not rewind the old or new files.

Remarks: Same as LIBEDIT,NR.

*OLD oldfile

Specify the name of the old file.

Remarks: Same as LIBEDIT,P=.

*RENAME rid,name

Rename a record.

Parameters: rid - the name of the replacement or old file record to be renamed

name - the new name

*REPLACE gid1,gid2,...

Replace old file records with records from the replacement file.

Examples: The old and replacement files each contain records A, B, C, D. To replace only C and D, use either of the following:

```
*FILE    replfyl          *FILE   replfyl
*NOREP   replfyl          *IGNORE A-B
*REPLACE C-D
```

*REWIND lfn

Rewind a file before and after editing.

*TYPE type (*NAME)

Set the default record type.

Parameters: type - the record type (ABS, OPL, OVL, PROC, REL, TEXT, ULIB)

*VFYLIB Verify the new file against the old file using VFYLIB.

Remarks: Overridden by *LIBGEN.

*** DTRC Object Libraries ***

The following object libraries have been added to NOS at DTRC:

```
DTLIB/UN=NSYS   - Subprograms written or maintained by the
                  Computer Center

                  To use: ATTACH,DTLIB/UN=NSYS.
                          LDSET,LIB=DTLIB.  -or-  LIBRARY,DTLIB.
                          LGO.

UTILITY/UN=NSYS - Programs written or maintained by the
                  Computer Center

                  To use: ATTACH,UTILITY/UN=NSYS.
                          LIBRARY,UTILITY.
                          prognam.
```

*** Examples ***

1) Create a library of subprograms.

```
ATTACH,mysubss.
FTN5,I=mysubss,OPT=2,L=0.
PURGE,mysubs/NA.
DEFINE,mysubs/CT=PU.
LIBGEN,P=mysubs.
```

2) Create a library of all subprograms from an UPDATE library.

```
ATTACH,OLDPL=mypl.
UPDATE,F.
FTN5,I,L=out2,OPT=2.
PURGE,mysubs/NA.
DEFINE,mysubs/CT=PU.
LIBGEN,P=mysubs.
ROUTE,out2,DC=PR.
```

3) Add a subprogram to an existing library and have the output list in alphabetical order.

```
Direct files                               Indirect files
------------                               --------------
jobnam3.                                   jobnam3.
USER,user,pw.                              USER,user,pw.
CHARGE,....                                CHARGE,....
FTN5,OPT=2.                                FTN5,OPT=2.
ATTACH,subs.                               GET,subs.
PURGE,NEW/NA.
DEFINE,NEW.
LIBEDIT,P=subs.                            LIBEDIT,P=subs.
PURGE,subs.                                REPLACE,NEW=subs.
CHANGE,subs=NEW/CT=PU.
<eor>                                      <eor>
*ADD LIB1,LGO                              *ADD LIB1,LGO
```

4) Delete subprogram BADSUB from an existing library.

```
GET,OLD=subs.
LIBEDIT,B=0,Z.*DELETE REL/badsub
REPLACE,NEW=subs.
```

***** Loader *****

The loader is responsible for loading all programs, resolving any external references, and optionally initiating execution.

Once loading of a program is started, no other control statements may interrupt the load sequence. For instance, a 'LOAD,lfn.' statement may only be followed by another 'LOAD,lfnl.' or one of the loader control statements or MAP, REDUCE or others listed in the Loader Reference Manual.

*** Types of Loading ***

Loading differs according to whether the input is one or more object modules or a single memory image module. Loading of object modules can involve overlay or segment generation and can result in one or more memory image modules. A basic load results in one memory image (absolute) module.

Object module loading — One or more object modules are loaded, libraries are searched for the external references, addresses are adjusted, and a memory image module may be produced.

Memory image loading — This is a special case because no external linkage or address adjustment is required.

Basic loading — All object code is loaded at the same time, resulting in a single memory image module.

Segmentation

For large programs, segmentation should be used to divide the program into several memory image modules, called segments.

With segmentation, only those portions of the program needed at a given moment are in memory. Different memory image modules reside in the same area of memory at different times. Depending on execution requirements, different memory image modules are loaded dynamically.

Features:

-Segmentation allows any number of levels, limited only to a total of 4093 segments.

-After segments have been generated, their loading is automatic.

-References between segments may be upward or downward.

-At execution time, a resident program is loaded which loads the root segment. Thereafter, it loads the other segments as required.

Overlay generation

An overlay is a collection of executable programs which are called into memory at execution time, according to an overlay structure which is defined in the source code.

Overlay capsule generation

For large programs, overlay capsules may be used to divide the program into an absolute main program and one or more capsules which are loaded and unloaded by the user.

Because overlays and overlay capsules require statements in your program to cause the overlaying to take place, they are not recommended. Instead, use segmentation, which is controlled by directives external to your program.

*** Loader Control Statements ***

See Appendix D for the syntax of the following control statements used to load a program.

EXECUTE	Complete loading, fill unsatisfied references by system (and user) library search, generate load map and execute the program.
LDSET	Set any of several loader options for the current load only.
LGO	Load and execute the default compiler binary output file.
LIBLOAD	Load modules from specified library which contains the specified entry points.
LOAD	A list of files whose contents are to be loaded.
name	Load and execute binary program or procedure in local file <name>.
NOGO	Complete the loading of a program, including generating load map, but do not execute.
SATISFY	Satisfy unsatisfied externals prior to normal satisfaction at load completion.
SLOAD	Selectively load modules from local file <lfn>.

In addition, the following Loader-related control statements are also available:

LIBRARY	Specify a set of global libraries to be searched for externals and programs and the order in which the libraries are to be searched.
MAP	Specify the global default option for load maps.
REDUCE	Turn the reduce flag on or off.
RFL	Set field length for the next program execution.

*** Segmentation ***

To implement segmentation, a separate directive record is prepared to describe the tree structure. The modules will be loaded automatically as needed. Job field length is adjusted dynamically if the program has no blank common, has no level statements and is not in RFL mode.

All necessary Record Manager routines must be in the root segment. Other LOAD and LDSET statements follow the SEGLOAD statement:

```
SEGLOAD,I=lfndir,B=lfnabs,LO=lfnout.
```

** SEGLOAD Directives **

x TREE y

To define a tree structure. <y> may be comma-separated list of other trees (pre-defined), segments or names of individual subprograms to be assigned a common starting address.

x TREE f-(c,d)

To indicate branching of the tree use -, then all following items are enclosed in parentheses.

c INCLUDE a,b

To assign programs <a> and <b> to segment <c>. Copies of a routine may be in different segments.

c GLOBAL com1,com2

To establish named commons at desired segment. Reference name to left of directive must be defined by a previous directive.

c GLOBAL com1,com2-save

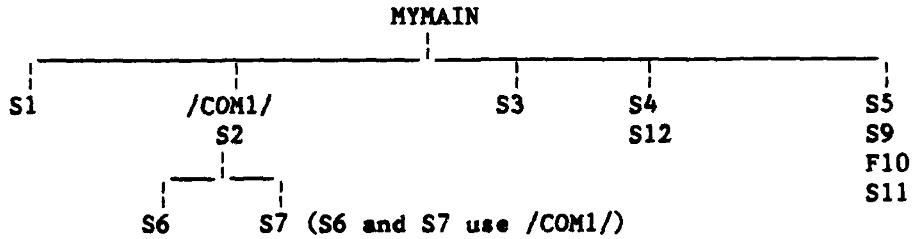
To save global block on disk for later calls to the segment which contains it.

END ept

Should be the last directive in the record, where <ept> is the entry of the main program in the root segment. Non-fatal error if omitted.

** Sample Tree Diagram **

A block data subprogram defines common /COM1/ which is to be loaded with program S2. /COM1/ is also referred to by S6 and S7.



(S6 and S7 use /COM1/)

```
SEG   INCLUDE  S2,BLKDAT.
SEG   GLOBAL   COM1
PLUM  TREE     SEG-(S6,S7)
PEAR  TREE     MYMAIN-(S1,PLUM,S3,S4,S5)
S5    INCLUDE  S9,F10,S11
S4    INCLUDE  S12
      END
```

By using nested parentheses one TREE directive may be eliminated.

*** Segmentation Cautions ***

1. To develop a segmented job, several runs may be required, so relocatable object code should be cataloged. Common blocks and Record Manager routines may need to be INCLUDEd in lower segments to operate properly.

2. The load map must be checked carefully for any duplicate common block entries. Each common block which is referenced in more than one segment must be put into a global at the nearest-to-the-root segment. If any common block appears more than once without "safe", a global is required to eliminate duplicate storage areas. If input/output is performed in several segments, some Record Manager common blocks may be multiply defined (e.g., AOB.RM or Q8.IO.).

Subfields in SEGLOAD directives which contain any of the special characters , - ( ) or which start with a $ must be defined as literals (i.e., delimited by $...$). Embedded $ is represented by $$, thus, $TAN must be specified as $$$TAN$.

When Record Manager common is global to a root segment, the loader may detect errors in initializing. If so, an INCLUDE directive will be required to move the RM routines to that segment (e.g., 'MYMAIN INCLUDE INCOM=').

3. Directives must not go beyond column 72 of a line. They may be broken almost anywhere and continued on the next one or more cards/lines. The continuations have a comma (,) in column 1 as the continuation signal, then the directive is continued starting in column 2.

Continued directives should be avoided, if possible, to improve readability.

4. FTN5 users should avoid passing external references as subprogram arguments. When the external is not in the root segment or the same segment as the call, execution will generate the fatal error message 'NON-EXECUTABLE WORD LOADING A SEGMENT'.

*** Compile, Load and Catalog Absolute Program ***

** Simple Load **

```
jobname,....
USER,user,pw.
CHARGE,....
FTN5,OPT=1.
LOAD,LGO.
NOGO,myprog.                      <-- absolute module into MYPROG
REPLACE,myprog.
<eor>
      program myprog
         ...
<eoi>
```

** SEGLOAD **

```
jobname,....                          name/code
USER,user,pw.
CHARGE,....
FTN5,OPT=1.
REPLACE,LGO=mysglgo.              <-- save relocatable modules for
                                      possible re-segmentation
PURGE,mysegnu/NA.                 <-- MYSEGNU for absolute segments
DEFINE,mysegnu.
SEGLOAD,B=mysegnu.
LOAD,LGO.
NOGO.                             <-- absolute segments onto MYSEGNU
PURGE,myseg1/NA.                  <-- replace old
CHANGE,myseg1=mysegnu.            <--  copy of MYSEGL
<eor>
   < FTN5 source program >
<eor>
   < SEGLOAD directives >
<eoi>
```

** Interactive Simple Execution **

```
GET,mysor.
FTN5,I=mysor,L=myout,OPT=1.
LDSET,MAP=S.                     <-- to see any missing routines
LGO.
...
ROUTE,myout,DC=PR.               <-- print the compilation listing
```

***** Other Software *****

*** Accessing Other Software ***

Programs obtained from other vendors are normally execute-only. To access them, you normally need

```
ATTACH,program/UN=un,M=E.      <-- A below
    -or-
GET,program/UN=un.             <-- G below
```

where un currently is APPLLIB, LIBRARY or NSYS.

*** UN=APPLLIB ***

As of the date of this page, UN=APPLLIB contains:

GPSS	A	General Purpose Simulation System
PERT78	A	Pert/Time

*** UN=LIBRARY ***

As of the date of this page, UN=LIBRARY contains:

BETONOS	A	NOS/BE to NOS command help (HELPBE) and NOS/BE file loader (BELOAD)
DTLIB	A	DTRC subprogram library
HOTSPOT	G	Analyze a program for inefficient code
PROCFIL	G	DTRC procedure library
SIMII5	A	Simscript II.5

*** UN=NSYS ***

As of the date of this page, UN=NSYS contains:

CALCFN	G	Calcomp Functional Package
CALC936	G	Calcomp 936 Subroutine Package (7-track tapes only)
DISSPLA	G	DISSPLA Graphic Subroutine Package
IMSLM	A	IMSL 10 Mathematical Subroutine Library
IMSLSS	A	IMSL 10 Special Function and Statistical Subroutine Library
LINPACK	G	Simultaneous Linear Algebraic Equation Solver Package
UTILITY	G	DTRC program library

***** Magnetic Tape *****

Magnetic tapes should be used for sequential data for such purposes as:

- . Transfer of information to and from other computers and off-line peripherals
- . Files which are used infrequently
- . Back-up copies of disk files
- . Long-term storage of data

Tapes should not be used for scratch files or random information. For safety, two copies on different tapes should be maintained, or for data which is updated, a grandfather-father-son system is advised. It is not wise to mount a tape containing good data, read through it, and write new data at the end. Instead, copy the existing data to a second tape and add the new data to the second tape, retaining the first tape as a back-up.

Processing a file on tape will take considerably more I/O time than on disk and more elapsed time.

Information concerning the physical and logical characteristics of the tape is specified in control statements.

Nine-track tapes are supported on the DEC VAX and CDC CYBER 860 computers; 7-track tapes are supported on the CDC CYBER 860. There are no tape drives on the Cray, so tapes must be accessed via one of the front ends.

*** Tape Labels ***

Tapes may be labelled or unlabelled. Labels should always be used except when writing data for, or reading data from a computer which cannot handle ANSI standard labels.

In general, a labelled tape has volume and end-of-volume labels, and may also have user labels. Each file on the tape may have its own header and trailer labels.

*** Tape Formats ***

Generally, records on tape are fixed or variable length, blocked or unblocked, ASCII or EBCDIC (9-track), BCD (7-track), coded, or binary. Where possible, tapes written by or for another computer should be 9-track, 1600 cpi, fixed length, blocked, ASCII.

*** Tape Care and Cleaning ***

Tapes should be stored in closed containers in racks which give them vertical support. Tapes may not be spliced. They should be read and rewound at least every six months. Logs should be kept on contents, format, and creation dates of tapes.

If a tape has many parity errors, cleaning it may help. Even a brand new tape may need cleaning. This off-line process does not destroy the information on the tape. If a tape receives heavy usage, cleaning it after ten or more uses may reduce the incidence of parity errors. A tape can also be certified, which determines whether there are any areas on the tape which do not record properly. Certification DESTROYS current information on the tape (except VSN). To change the VSN, contact the Tape Librarian and request blank labelling or degaussing.

If, after a tape has been cleaned, it still has many parity errors, it should be exchanged for a new tape. The information on the old tape is not recovered automatically in this case.

To have a tape cleaned or certified, submit an off-line work request to the Tape Librarian. Users who are not at the Carderock site should call (202) 227-1967.

When possible, slot tapes should be in the Computer Room environment for at least two hours before reading or writing. This allows temperature and humidity to stabilize and should minimize tape problems.

Please notify Code 1893.1, (202) 227-1907, of any unusual tape problems.

*** Using Tapes on the CYBER 860 ***

The CDC CYBER 860 has six 9-track tape drives (four for 6250/1600 cpi and two for 1600/800 cpi), and two 7-track tape drives (800/556 cpi).

The following control statements are used to access or analyze them:

LABEL Mount a magnetic tape and, if labelled, check the label.

LISTLB List the labels of an ANSI-labelled tape.

RESOURC Specify that more than one tape drive is required.

TDUMP Octal and alphanumeric dump of all or part of a file.

VSN Associate a local file name with one or more volume serial numbers.

** Examples **

The following examples illustrate tape usage in batch jobs. Tapes may also be used interactively (without the job, USER and CHARGE statements).

1. Unlabelled NOS/BE Tape to Disk

```
xxxx.             job statement.
USER,xxxx,upw.
CHARGE,1234567890.
DEFINE,disk/CT=PU.
LABEL,tape,F=SI,LB=KU,VSN=NA9999,D=1600,PO=R,R.
COPYBF,tape,disk,5.
UNLOAD,tape.
```

2. Copy Old Stranger (Foreign) Tape to New - 6250 multifile

```
xxxx.
USER,xxxx,upw.
CHARGE,1234567890.
RESOURC,GE=1.                    <-- one additional tape drive
LABEL,t4,VSN=NA9998,D=GE,PO=W,W,F=S,L=softwstr.
VSN,t5=SLOTxx=CA9995.
LABEL,t5,PO=R,R,F=S,D=GE,L=softwstr.
COPY,t5,t4,EL=10,M=coded,PO=E.
UNLOAD,t5.
```

*** Using Tapes on the DEC VAX ***

The DEC VAXcluster has six 9-track tape drives (6250/1600 cpi); the remote mini in Annapolis has two.

The following control statements are used to access or analyze them:

ALLOCATE Assign a tape drive to a logical name.

DEALLOCATE Return a previously allocated device and disassociate the job's logical name from the tape drive.

DISMOUNT Release a tape volume that was previously mounted.

INITIALIZE Initialize a magnetic tape.

MOUNT Mount a magnetic tape and, if labelled, check the label.

** Examples **

1. Initialize a VAX/VMS tape:

```
$! TAPINIT.COM : initialize VAX/VMS tape, default is 1600
$!
$    if p3 .nes. "1600" .and. p3 .nes. "6250" then p3 = "1600"
$!
$    allocate mu: tape                ! get any available tape drive
$!
$    mount tape: /foreign /density='p3' -
                /comment="mount slot''p1' vsn=''p2' ringin"
$    dismount tape /nounload
$    initialize tape 'p2'
$    deallocate tape
$    exit
$!
$! p1 - 1- or 2-digit slot number or NONE
$! p2 - 6-character VSN
$! p3 - density (6250 or 1600) defaults to 1600
$!
$! created 06/23/88 by CASG
$! last modified 06/24/88 @ 1146 by CASG (add "?" for help)
$!
$! End of TAPINIT.COM
```

The above is a portion of the actual procedure to show just the defaulting of density and how to initialize a tape. To see the full procedure, which includes validation of each parameter, and allows "?" for help for the procedure and each parameter, type "TYPE VSYS:TAPINIT.COM".

***** Conversion to the Network *****

*** Fortran Considerations ***

Fortran 66 (FTN/FTN4 on the CDC CYBER 750) programs should be converted to Fortran 77 prior to moving to the network.

The Fortran 77 (FTN5) compiler on the CDC CYBER 860A is the same as on the CDC CYBER 750, so no conversion is needed. Programs converted from CDC or the VAXcluster to another computer should not need modification if no extensions are used. Extensions, of course, will require manual change.

*** Cobol Considerations ***

Cobol is not available on the Cray.

Cobol 68 (COBOL 4 on the CDC CYBER 750) is not available on the VAXcluster or the CDC CYBER 860A. Such programs should be converted to Cobol 74 prior to moving to the network.

The Cobol 74 (COBOL5) compiler on the CDC CYBER 860A is the same as on the CDC CYBER 750, so no conversion is needed. Programs converted from CDC or the VAXcluster to the other computers should not need modification if no extensions to the Cobol Standard are used. Extensions, of course, will require manual change.

***** Appendix A *****

*** ASCII Character Set ***

char	ASCII (hex)	EBCDIC (hex)	Display (octal)	char	ASCII (hex)	EBCDIC (hex)	Display (octal)
NUL	00	00		(((	28	4D	51
SOH	01	01		)))	29	5D	52
STX	02	02		***	2A	5C	47
ETX	03	03		+++	2B	4E	45
EOT	04	37		,,,	2C	6B	56
ENQ	05	2D		---	2D	60	46
ACK	06	2E		...	2E	4B	57
BEL	07	2F		///	2F	61	50
BS	08	16		000	30	F0	33
HT	09	05		111	31	F1	34
LF	0A	25		222	32	F2	35
VT	0B	0B		333	33	F3	36
FF	0C	0C		444	34	F4	37
CR	0D	0D		555	35	F5	40
SO	0E	0E		666	36	F6	41
SI	0F	0F		777	37	F7	42
DLE	10	10		888	38	F8	43
DC1	11	11		999	39	F9	44
DC2	12	12		:::	3A	7A	63
DC3	13	13		;;;	3B	5E	77
DC4	14	3C		<<<	3C	4C	72
NAK	15	3D		===	3D	7E	54
SYN	16	32		>>>	3E	6E	73
ETB	17	26		???	3F	6F	71
CAN	18	18		@@@	40	7C	74
EM	19	19		AAA	41	C1	01
SUB	1A	3F		BBB	42	C2	02
ESC	1B	27		CCC	43	C3	03
FS	1C	1C		DDD	44	C4	04
GS	1D	1D		EEE	45	C5	05
RS	1E	1E		FFF	46	C6	06
US	1F	1F		GGG	47	C7	07
space	20	40	55	HHH	48	C8	10
!!!	21	4F	66	III	49	C9	11
"""	22	7F	64	JJJ	4A	D1	12
###	23	7B	60	KKK	4B	D2	13
$$$	24	5B	53	LLL	4C	D3	14
%%%	25	6C		MMM	4D	D4	15
&&&	26	50	67	NNN	4E	D5	16
'''	27	7D	70	OOO	4F	D6	17

char	ASCII (hex)	EBCDIC (hex)	Display (octal)
PPP	50	D7	20
QQQ	51	D8	21
RRR	52	D9	22
SSS	53	E2	23
TTT	54	E3	24
UUU	55	E4	25
VVV	56	E5	26
WWW	57	E6	27
XXX	58	E7	30
YYY	59	E8	31
ZZZ	5A	E9	32
[[[	5B	4A	
\\\	5C	E0	75
]]]	5D	5A	
^^^	5E	5F	76
___	5F	6D	65
grave	60	79	
aaa	61	81	
bbb	62	82	
ccc	63	83	
ddd	64	84	
eee	65	85	
fff	66	86	
ggg	67	87	

char	ASCII (hex)	EBCDIC (hex)	Display (octal)
hhh	68	88	
iii	69	89	
jjj	6A	91	
kkk	6B	92	
lll	6C	93	
mmm	6D	94	
nnn	6E	95	
ooo	6F	96	
ppp	70	97	
qqq	71	98	
rrr	72	99	
sss	73	A2	
ttt	74	A3	
uuu	75	A4	
vvv	76	A5	
www	77	A6	
xxx	78	A7	
yyy	79	A8	
zzz	7A	A9	
{{{	7B	C0	61
¦¦¦	7C	6A	
}}}	7D	D0	62
˜˜˜	7E	A1	
DEL	7F	07	

*** CDC Character Set ***

Display	char-	punch	punch	7-track	9-track		note/name
Code	acter	026	029	ext BCD	ASCII	EBCDIC	
			if				
			diff				
00	:::	2-8			25	6C	colon (1,2)
01	AAA	12-1		61	41	C1	
02	BBB	12-2		62	42	C2	
03	CCC	12-3		63	43	C3	
04	DDD	12-4		64	44	C4	
05	EEE	12-5		65	45	C5	
06	FFF	12-6		66	46	C6	
07	GGG	12-7		67	47	C7	
10	HHH	12-8		70	48	C8	
11	III	12-9		71	49	C9	
12	JJJ	11-1		41	4A	D1	
13	KKK	11-2		42	4B	D2	
14	LLL	11-3		43	4C	D3	
15	MMM	11-4		44	4D	D4	
16	NNN	11-5		45	4E	D5	
17	OOO	11-6		46	4F	D6	
20	PPP	11-7		47	50	D7	
21	QQQ	11-8		50	51	D8	
22	RRR	11-9		51	52	D9	
23	SSS	0-2		22	53	E2	
24	TTT	0-3		23	54	E3	
25	UUU	0-4		24	55	E4	
26	VVV	0-5		25	56	E5	
27	WWW	0-6		26	57	E6	
30	XXX	0-7		27	58	E7	
31	YYY	0-8		30	59	E8	
32	ZZZ	0-9		31	5A	E9	
33	000	0		12	30	F0	(sometimes 00)
34	111	1		01	31	F1	
35	222	2		02	32	F2	
36	333	3		03	33	F3	
37	444	4		04	34	F4	
40	555	5		05	35	F5	
41	666	6		06	36	F6	
42	777	7		07	37	F7	
43	888	8		10	38	F8	
44	999	9		11	39	F9	
45	+++	12	12-6-8	60	2B	4E	plus
46	---	11		40	2D	60	minus
47	***	11-4-8		54	2A	5C	asterisk
50	///	0-1		21	2F	61	slash
51	(((	0-4-8	12-5-8	34	28	4D	left paren
52	)))	12-4-8	11-5-8	74	29	5D	right paren
53	$$$	11-3-8		53	24	5B	dollar
54	===	3-8	6-8	13	3D	7E	equal
55				20	20	40	blank
56	,,,	0-3-8		33	2C	6B	comma
57	...	12-3-8		73	2E	4B	period

Display	char-	punch	punch	7-track	9-track		note/name
Code	acter	026	029	ext BCD	ASCII	EBCDIC	
			if				
			diff				
60	###	0-6-8	3-8	36	23	7B	pound
61	[[[	7-8	12-2-8	17	5B	4A	l bracket
62	]]]	0-2-8	11-2-8	32	5D	5A	r bracket
63	%%%	2-8			25	6C	percent (1,2)
63	:::	2-8			25	6C	colon (1,2) (sometimes 16)
64	"""	4-8	7-8	14	22	7F	quote
65	___	0-5-8		35	5F	6D	underline
66	!!!	11-2-8	12-7-8	52	21	4F	exclam (3)
66	!!!	11-0		52	21	4F	exclam (3)
67	&&&	0-7-8	12	37	26	50	ampersand
70	'''	11-5-8	5-8	55	27	7D	apostrophe
71	???	11-6-8	0-7-8	56	3F	6F	question
72	<<<	12-2-8	12-4-8	72	3C	4C	less than (3)
72	<<<	12-0		72	3C	4C	less than (3)
73	>>>	11-7-8	0-6-8	57	3E	6E	greater than
74	@@@	5-8	4-8	15	40	7C	at
75	\\\	12-5-8	0-2-8	75	5C	E0	reverse slant
76	^^^	12-6-8	11-7-8	76	5E	5F	circumflex
77	;;;	12-7-8	11-6-8	77	3B	5E	semicolon (4)
55		6-8	0-4-8		20	40	blank (5)

Notes:

(1) In the 63-character set (NOS/BE), Display Code 00 has no character, and 63 is the colon (:). In the 64-character set (NOS), 00 is the colon (:), and 63 is the percent (%).

(2) On 7-track tape, this becomes zero (display 33).

(3) Alternate punches.

(4) Avoid a whole word of semicolons, which is a negative zero and is treated as an end-of-record.

(5) On some terminals, this is transmitted as a binary zero. For these terminals, avoid putting this punch in columns 9-10, 19-20, ..., 79-80, as each will be interpreted as a zero-byte terminator.

***** Appendix B *****

*** Cray JCL Commands ***

Cray JCL commands have the following general syntax:

```
verb sep1 param1 sep2 param2 ... sepn paramn term comments
```

```
verb     is the name of the routine to be executed.  It consists of an
         alphabetic character (A-Z, a-z, $, %, @) followed by 0-6
         alphanumeric characters for system, local dataset name and
         system dataset name verbs; or 1-8 alphanumeric characters for
         library-defined verbs.

sepi     are separators and include:
          ,      - VERB,parameter.
          (      - VERB(parameter).

          .      - VERB,parameter.     <-- use period if comma
          )      - VERB(parameter)     <-- use right paren if left paren

          ,      - VERB(parameter,parameter)

          =      - VERB(keyword=value)

          :      - VERB(keyword=value1:value2)

          ^      - VERB(...parameters...^      <-- statement continued
                   parameters)                 <-- on another line

          '...' - VERB(keyword='string')

          (...) - VERB(keyword=(value:value))

parami   are parameters, which may be positional or keyword.
         Positional parameters have one of the following formats:
             value
             value1:value2:...:valuen
         Keyword parameters have one of the following formats:
             keyword
             keyword=value
             keyword=value1:value2:...:valuen

term     is the statement terminator.  It is either a period
             VERB.
             VERB,parameters.
         or a right parenthesis
             VERB(parameters)

comments follow the terminator.
```

*** Strings ***

The following string representations are used in this appendix:

```
aa...a    1 or more alphabetic characters

axx...x   1 or more alphanumeric characters, the first alphabetic

xxx...x   1 or more alphanumeric characters

nnn...n   1 or more decimal (unless otherwise stated) digits
```

*** Some Common Parameters ***

The following parameters are used in many JCL commands. If they have a different meaning or a special condition, it will be mentioned in the individual description.

```
AM=mode      Alternate User Access Mode (see PAM=)

DC=dc        Disposition code
              IN - input queue of destination station
              MT - magnetic tape at job origin mainframe
              PR - print at job origin mainframe
              SC - scratch the dataset
              ST - stage to mainframe (make permanent at job origin
                   mainframe)

DF=df        Dataset format (blocking; front-end conversion)
              BB - binary blocked      (no reblocking, no conversion;
                                        for object modules, graphics
                                        output, etc.)
              BD - binary deblocked    (same as TR)
              CB - character blocked   (front-end converts to ASCII
                                        (VAX) or Display Code (NOS))
              CD - character deblocked (front-end converts to ASCII
                                        (VAX) or Display Code (NOS))
              TR - transparent         (no deblocking; no conversion)
             (default: CB)

DN=dn        Local dataset name        (axxxxxx, 7 maximum)

ED=ed        Edition number (1-4095)

ERR          Suppress error termination messages

EXO=exo      Execute option
              ON  - execute-only (cannot be read or PSDUMPed)
              OFF - not execute-only

I=idn
IDN=idn      Input dataset name        (normal default: $IN)

ID=uid       Additional permanent dataset ID
                                       (axxxxxxx, 8 maximum)
```

L=ldn	Name of dataset to contain the listing (default: $OUT)
M=mn	Maintenance control word (axxxxxxx, 8 maximum)
MF=mf	Front-end computer N1 - CDC CYBER 180/860A (NOS) V3 - DEC VAXcluster node DT3 (VMS) (default: front-end of job origin)
MSG	Suppress normal termination messages
NA	No abort. If omitted, an error causes the job step to abort.

O=odn
ODN=odn
	Output dataset name (normal default: $OUT)

OWN=owner	Owner of the permanent dataset (not needed for your own files)
PAM=mode	Public Access Mode E - execute only (same effect as EXO=ON) M - maintenance only N - no public access R - read only W - write only Example: PAM=R:W gives read and write permission (default: N)
PDN=pdn	Permanent dataset name (xxxxxxxxxxxxxxx, 15 maximum; enclosed in quotes "..." if other than A-Z,0-9)
R=rd	Read control word (axxxxxxx, 8 maximum)
TEXT='text'	Text (up to 240 character) to be passed to the front-end, enclosed in apostrophes ('...')
TID=tid	Destination terminal (default: terminal of job origin)
UQ	Unique access (required to delete or modify a dataset) (default: multiple access)
W=wt	Write control word (axxxxxxx, 8 maximum)

*** Permanent Dataset Utility Shorthand Notation ***

In the permanent dataset utility commands, wildcards may be used in the PDN, PDS, ID, US, and OWN parameters. An asterisk "*" represents any single character; a minus sign "-" represents zero or more characters. They are illustrated with PDN=.

PDN=ABC- all permanent dataset names starting with ABC

PDN=A*** all 4-character permanent dataset names starting with A

PDN=-A*- all permanent dataset names containing the letter A followed by one or more other characters

PDN=- all permanent dataset names

PDN=***- all permanent dataset names having 3 or more characters

*** A Word About Continuations ***

If a COS JCL statement is too long to fit on one line, it may be continued by breaking the statement after a parameter, ending the line with a carat (^), and continuing the statement on the next line(s). For example,

```
FETCH,DN=prog3,SDN=myprog,^
      TEXT='GET,myprog.CTASK.'.
```

If a text field (quoted string) is too long, it may be split anywhere by adding an apostrophe (') to close the partial string and a carat to end the first line, and starting the next line with an apostrophe immediately followed by the rest of the string. For example,

```
DISPOSE,DN=FT14,SDN=myout14,DC=ST,MF=N1,TEXT='USER,user,pw.'^
        'PURGE,myout14/NA.DEFINE,myout14.CTASK.'.
```

-or-

```
DISPOSE,DN=FT14,SDN=myout14,DC=ST,MF=N1,^
        TEXT='USER,user,pw.'^
             'PURGE,myout14/NA.'^
             'DEFINE,myout14.'^
             'CTASK.'.
```

*** Summary of Cray JCL Commands ***

The following are Cray JCL statements, except as indicated by:

```
(DTRC - x)  A command, procedure or program added at DTRC.  Unless
            otherwise noted, these are accessed by:
                ACCESS,DN=x,OWN=PUBLIC.
                LIBRARY,DN=x:*.
                name,....
            x is one of: PROCLIB, UTILITY.


*        Entire line is a comment.

         Syntax:      *  <comments>

         Similar commands:  NOS/BE:  COMMENT
                            NOS:     COMMENT; *
                            VMS:     !

         Examples:    * This is a comment ---

ACCESS   Make a permanent dataset local.

         Syntax:      ACCESS,DN=dn,PDN=pdn,ID=uid,ED=ed,R=rd,W=wt,M=mn,
                            UQ,NA,ERR,MSG,OWN=owner.


         Parameters:  PDN=pdn - If omitted, dn is used.

                      R=rd    - required to read the dataset if R= on
                                SAVE

                      W=wt    - required for ADJUST if W= on SAVE

                      M=mn    - required to DELETE the dataset if M=
                                on SAVE

         Similar commands:  NOS/BE:  ATTACH
                            NOS:     ATTACH; GET
                            VMS:     no local file concept

         Examples:    ACCESS,DN=mylocal,PDN=mypermfile.
                      ACCESS,DN=mylcl,PDN=yourpermfile,OWN=yourid.
                      = = = = =
                      ACCESS,DN=myfile,UQ.
                      DELETE,DN=myfile.


ACCOUNT  Validate the user.  Follows the JOB statement or, is the first
         interactive statement.

         Syntax:      ACCOUNT,AC=ac,US=us,UPW=upw,NUPW=nupw.
```

Parameters: AC=ac - Account number (required)
(10 digits or "S" + 9 digits)

US=us - Username (your 4-character User Initials)

UPW=upw - User password (required)

NUPW=nupw - New user password

Remarks: This must be the first statement of an interactive session. When entered via CDC NOS ICF, US= may be omitted because it is supplied automatically. When entered via the DEC VMS Cray Station, US= may be omitted if you entered it in upper case in response to the CRAY USERNAME: prompt.

See also: JOB

Similar commands: NOS/BE, NOS: CHARGE
VMS: no user-specified charging

Examples:

```
ACCOUNT,AC=1234567890,US=xxxx,UPW=mypass.
ACCOUNT,AC=1234567890,US=xxxx,
      UPW=mypass,NUPW=nupass.
```

ACQUIRE Get a front-end dataset and make it local and permanent.

Syntax:

```
ACQUIRE,DN=dn,PDN=pdn,AC=ac,ID=uid,ED=ed,RT=rt,
      R=rd,W=wt,M=mn,UQ,MF=mf,TEXT='text',DF=df,
      OWN=ov,PAM=mode,ERR,MSG.
```

Parameters: AC=ac - acquisition code
IN - input dataset
IT - intertask communication
ST - dataset staged from front end (MF=)
(default: ST)

ED=ed - (defaults: 1 (permanent dataset does not exist)
highest (permanent dataset exists))

RT=rt - retention period (1-4095 days)
(default: 45)

Remarks: If the dataset is permanent, ACQUIRE is the same as ACCESS. If not, then it is the same as FETCH, SAVE,ACCESS.

See also: FETCH, MSFETCH

```
        Similar commands:  NOS/BE:  MSFETCH
                           NOS:     ATTACH; GET
                           VMS:     HFT FETCH

        Examples:     ACQUIRE,DN=myfile,PDN=myfile,TEXT='myfile.FOR'.


ADJUST  Redefine size of a permanent dataset.

        Syntax:       ADJUST,DN=dn,NA,ERR,MSG.

        Permissions required: write; UQ on ACCESS

        Remarks:      ADJUST attempts to close the file.  Subsequent
                      references in the same job must reopen it and
                      begin at BOD.

        Similar commands:  NOS/BE:  ALTER; EXTEND
                           NOS:     APPEND
                           VMS:     lengthened automatically; cannot
                                    be shortened

        Examples:     ADJUST,DN=myfile,NA.


ALTACN  Validate an alternate account number for permanent files.

        Syntax:       ALTACN,AC=ac.

        Parameters:   ac - the alternate account number

        Remarks:      ALTACN validates the supplied Job Order Number.

                      To use the validated number, specify the ACN
                      parameter on the SAVE or MODIFY command.

        See also:     MODIFY, SAVE

        Similar commands:  NOS:     CHANGE
                           NOS/BE:  RENAME

        Examples:     ALTACN,AC=1222233344.      <-- define the number
                      ...
                      SAVE,DN=newfyl,ACN.        <-- use the number
                      MODIFY,DN=oldfyl,ACN.      <-- change the number


ASSIGN  Create a local dataset and assign dataset characteristics.

        Syntax:       ASSIGN,DN=dn,LM=lm,A=alias,BS=bs,U.

        Parameters:   LM=   - maximum number of 512-word blocks in the
                              dataset
                              (default: 100000)
```

A= - alternate unit name

BS= - octal number of 512-word blocks for the I/O buffer
(default: 10 octal)

U - unblocked dataset
(default: blocked)

Remarks: See COS Reference Manual for additional parameters.

At system initiation,

```
   ASSIGN,DN=$IN,A=FT05.
   ASSIGN,DN=$OUT,A=FT06.
```

are performed automatically. You may reassign them at any time.

A Fortran OPEN will not recognize an ASSIGNed dataset.

Similar commands: NOS/BE: REQUEST
NOS, VMS: ASSIGN

Examples:

```
ASSIGN,DN=myinput,A=FT11.
                  ^-- Fortran program reading from
                      unit 11 will read file MYINPUT
                      instead
```

AUDIT Report on permanent datasets.

Syntax:

```
AUDIT,L=ldn,PDN=pdn,ID=uid,OWN=own,ACN=acn,
      LO=opt:...:opt,SZ=dsz,ACC=opt:opt,
      X=mm/dd/yy:'hh:mm:ss',
      TCR=mm/dd/yy:'hh:mm:ss',
      TLA=mm/dd/yy:'hh:mm:ss',
      TLM=mm/dd/yy:'hh:mm:ss'.
```

Parameters: L= - list dataset name
(default: $OUT)

PDN= - name of permanent dataset(s) to be listed

ID= - list datasets with this ID
ID - list datasets with null ID

OWN= - list datasets with this ownership value

ACN= - list datasets with this account number

```
LO=    - list options:
          S - short list (PDN, ID, ED; 2 per line)
              (may not be mixed with other options)
          A - access tracking (owner name, count,
              time of last and first accesses)
          B - backup info (backup volume name,etc.)
          L - long list (PDN, ID, ED, size (words),
              retention time, access count, track
              access flag, public access mode
              (PAM), creation, last access, last
              dump time, device name, preferred
              residency (PR), current residency
              (CR).
              (default in batch if no LO)
          N - notes list
          P - permit list (permitted owner name,
              access mode, access count, time of
              last access, time of permit creation)
          R - retired dataset list (same as L, but
              only retired datasets)
          T - text list
          X - extended long list (L plus number of
              blocks and words allocated)

SZ=    - list datasets >= this size (in words)

ACC=   - access option parameters
          AM  - those datasets belonging to OWN
                that you are allowed to see
          PAM - those datasets belonging to OWN
                having any form of public access
                (R:W:M:E)

X=     - list datasets expired as of this date
X      - list datasets expired as of now

TCR=   - list datasets created since this date
TCR    - not allowed
         TCR=mm/dd/yy is sufficient

TLA=   - list datasets not accessed since this date
TLA    - not allowed
         TLA=mm/dd/yy is sufficient

TLM=   - list datasets modified since this date
TLM    - not allowed
         TLM=mm/dd/yy is sufficient
```

```
Similar commands:  NOS/BE:  AUDIT; BEGIN,AUDIT
                   NOS:     CATLIST
                   VMS:     DIRECTORY; MSSAUDIT
```

```
Examples:     AUDIT,LO=S           <-- short audit
              AUDIT,LO=P           <-- audit showing who can and
                                       has accessed the datasets
              AUDIT,LO=L:P:N       <-- long audit, permitted
                                       users and notes
              AUDIT,LO=L           <-- long audit
              AUDIT,OWN=PUBLIC.    <-- list public files
```

AUDPL   Audit an UPDATE program library (PL).

```
Syntax:        AUDPL,P=pdn,I=idn,L=ldn,M=mdn,*=m,/=c,DW=dw,
                    LW=lw,JU=ju,DK=list,PM=list,LO=string,
                    CM,NA,NR.

Parameters:  P  I  L  *  /  NR  - see UPDATE

             M=  - Modifications dataset name (will contain
                   reconstructed modification sets)
                   (default: $MODS)
             M=0 - No modifications output

             DW= - Data width (number of characters written
                   per line to M dataset)
                   (default: up to DW value on UPDATE stmt)

             LW= - Listing width (number of characters written
                   per line to L dataset
                   (Values: divided into pages:  80,  132;
                            continuous listing: C80, C132)
                   (default: 132, divided into pages)

             JU= - Justification
                     N - identifier name left-justified;
                         sequence number right-justified;
                         no period between
                     L - entire sequence field left-justified
                         with period between
                   (default: identified name right-justified;
                             sequence number preceded by a
                             period and left-justfied)

             DK=dk1:dk2:...:dkn                                (1)
             DK='dk1,dk2,...,dkj.dkk,...,dkn'                  (2)
                 - Decks for A, C, D, H, I options and PM
                   parameter
                    (For (1): up to 100 decks;
                     for (2): separate single decks with
                              commas, and ranges of decks with
                              periods)
                   (Maximum string length: 96 characters)
                   (default: options apply to all decks)
             DK  - By itself is invalid
```

```
                PM=id1:id2:...:idn                          (1)
                PM='id1,id2,...,idj.idk,...,idn'            (2)
                    - Pulled modification sets (reconstructs
                      modification sets for the listed
                      identifiers for the decks listed in DK)
                       (Syntax: same as for DK=)
                PM  - By itself is invalid

                LO=string
                    - Listing options for ldn
                       Text listing (for DK= decks, if specified)
                        A - active lines
                        C - conditional text directives
                            (subset of option D)
                        D - compile dataset generation directives
                            (subset of option A)
                        H - modification histories
                        I - inactive lines
                       Summary options (for the entire PL)
                        K - deck line counts
                        L - identifier list
                        M - modification set cross-reference
                        N - identifier list in ASCII order
                        O - overlapping modification set list
                        P - short summary of the PL
                        S - status of modification set
                        X - common deck cross-reference

                CM  - Copy modifications (reconstructed
                      modification sets) to ldn and mdn

                NR  - Do not rewind modifications or binary
                      identifier list datasets at start or end
                      of AUDPL

         Similar commands:  NOS/BE, NOS:  UPDATE
                            VMS:          CMS; LIBRARIAN;
                                          INCLUDE (in FORTRAN)

         Examples:      AUDPL,P=mypl,LO=P.
                        = = = = =
                        AUDPL,P=mypl,PM=mod2a:mod3c:example,
                              LO=AIKLMNOPSX.
                        COPYF,I=$MODS.


BLOCK    Convert an unblocked dataset to a blocked dataset.

         Syntax:        BLOCK,DN=ldn,BLKSIZE=size.          (1)

                        BLOCK,I=idn,O=odn,BLKSIZE=size.     (2)

         Parameters:    DN=      - the dataset to be replaced (using an
                                   intermediate dataset $UNBLK)
                                   (ldn is rewound before and after)
```

BLKSIZE= - record length in 64-bit words
(non-foreign datasets only)
((2) - not permitted if previously assigned as foreign; record length and type are taken from the input ASSIGN)

I= - the unblocked input dataset
(idn is not rewound before the copy)

O= - the blocked output dataset
(if previously opened (ASSIGN), odn is not rewound before; otherwise, odn is created)

Remarks: For foreign datasets, the record length and type are taken from the ASSIGN.

BLOCK is intended primarily for postprocessing datasets created by or for certain stations.

Examples:

```
BLOCK,DN=myfile.
             ^-- Replace MYFILE with blocked copy
                 of itself
= = = = =
BLOCK,I=myunblk,O=myblk.
             ^-- Copy unblocked file MYUNBLK as
                 blocked file MYBLK
```

BUILD Generate and maintain library datasets.

Syntax:

```
BUILD,I=idn,L=ldn,OBL=odn,B=bdn,NBL=ndn,
      SORT,NODIR,REPLACE.
```

Parameters:

I=idn - Directive dataset name
(default: $IN)
I - Same as I=$IN
I=0 - No directives

L=ldn - List dataset name
(default: $OUT)
L - Same as L=$OUT

OBL=odn - Old object library dataset name
(default: $OBL)
OBL - Same as OBL=$OBL
OBL=0 - No old binary library

B=bdn - Dataset with new object modules
(default: $BLD)
B - Same as B=$BLD
B=0 - No modules to be added

NBL=ndn - Output new object library dataset name (default: $NBL)

NBL - Same as NBL=$NBL

NBL=0 - No output written

SORT - modules are to be output in alphabetical order
(default: written in the order they were first read)

NODIR - Do not append the directory to the output dataset
(default: append the directory)

REPLACE - Modules in the new library are replaced and in the same order as in the old library
(default: new modules follow the unreplaced modules in the new library)

Directives: A directive consists of a keyword and, perhaps, a comma-separated list of dataset or module names. The keyword is separated from its list by a blank. Directives cannot be continue. Multiple directives, separated by a semicolon or period, may appear in one line.

FROM dn1,dn2,...,dnn
Single dataset for COPY, OMIT, LIST, or a list of datasets (copy dn1 thru dnn-1 to $NBL, dnn is the same as if specified alone. If no COPY, OMIT, dnn is also copied. dni can be a library or sequential dataset (like $BLD).

OMIT fn1,fn2,...,fnn
List of modules to be excluded. Each fni may be a single name or a group name, i.e., with wildcards - (any 0 or more characters) or * (any single character).

COPY fn1,fn2,...,fnn
List of modules to be copied. Each fni may be a single or group name, or a rename (ELM=OAK copies ELM and renames it OAK), or an inclusive range such as (first,last) or (first,) or (,last) or (,).

LIST
Immediately list characteristics of modules in input dataset.

See also: Section 2-6

```
Similar commands:  NOS/BE:  EDITLIB
                   NOS:     LIBEDIT
                   VMS:     LIBRARIAN

Examples:     BUILD,OBL=0,I=0.
              SAVE,DN=$NBL,PDN=mylib.
                          ^-- create a new library from $BLD
              = = = = =
              ACCESS,DN=$OBL,PDN=mylib.
              BUILD,I=0.
              SAVE,DN=$NBL,PDN=mylib.
                          ^-- add modules from $BLD to
                              existing library
              = = = = =
              ACCESS,DN=mylib1.
              ACCESS,DN=mylib2.
              ACCESS,DN=mylib3.
              BUILD,I,OLB=0,B=0.
              SAVE,DN=$NBL,PDN=mylib4.
              - Directive: FROM mylib1,mylib2,mylib3
                          ^-- merge several libraries - if
                              duplicate module names, last
                              found is retained (or use rename
                              form, if desired)
              = = = = =
              ACCESS,DN=$OBL,pdn=mylib.
              BUILD,B=0.
              SAVE,DN=$NBL,PDN=mylib.
              - Directive: OMIT badpgm
                          ^-- remove a module from a library
              = = = = =
              ACCESS,DN=xyz,PDN=mylib.
              BUILD,I,OBL=xyz,B=0,NBL=$BLD,NODIR.
              - Directive: COPY myprog
                          ^-- extract module for loading
```

CALL   Read control statements from the first file of another dataset or transfer control to a procedure.

```
Syntax:       CALL,DN=dn.          <-- read from another file
              CALL,DN=dn,CNS.      <-- call a procedure

Parameters:   DN=dn - the dataset containing the statements or
                      procedure (rewound before use)

              CNS   - Crack Next Statement - the first statement
                      in "dn" is the procedure header; the
                      statement following the CALL is treated
                      as the invocation of the procedure

See also:     Section 2-3

Similar commands:  NOS/BE, NOS:  BEGIN
                   VMS:          @name
```

Examples: Without CNS:

If the first file of dataset XYZ contains:

```
ACCESS,DN=INFYL,PDN=MYFILE.
ACCESS,DN=FILE1,PDN=MYDATA.
```

Then CALL,DN=XYZ. will access both datasets. This might be useful if you have several jobs using the same files, or if you have the same processing to be done by many jobs.

With CNS:

If the first file of dataset XYZ contains:

```
G,FILE,DATA.
ACCESS,DN=INFYL,PDN=&FILE.
ACCESS,DN=FILE1,PDN=&DATA.
```

Then

```
CALL,DN=XYZ,CNS.
    *,MYFILE,MYDATA
```

will access the datasets MYFILE and MYDATA. Note that PROC and ENDPROC statements and the procedure name (G) are not used.

"call by name"

Execute a program by its local file name.

Syntax:

```
dn.
dn,parameters.
```

Parameters: depends upon the local file being executed

```
Similar commands:  NOS/BE, NOS:  LGO  or  an lfn
                   VMS:          $ name :== $ dir:name
                                 $ name
```

Examples:

```
ACCESS,DN=myobj.
myobj.
```

CFT Compile a Fortran source program.

Syntax

```
CFT,I=idn,L=ldn,B=bdn,C=cdn,E=m,EDN=edn,
    OPT=option,MAXBLOCK=mb,INT=il,ALLOC=alloc,
    ON=string,OFF=string,TRUNC=nn,AIDS=aids,
    CPU=cpu:hdw,UNROLL=r,LOOPMARK[=lmmsgs],
    DEBUG,SAVEALL,ANSI.
```

Parameters: I= - Input dataset name (default: $IN)

L= - Listable output (default: $OUT)

L=0 - List only fatal errors

B= - Binary load module dataset name (default: $BLD)

B=0 - No binary load modules

C= - pseudo-CAL output dataset name (default: no dataset)

E= - Highest level of messages to be suppressed

- 1 - comment
- 2 - note
- 3 - caution
- 4 - warning
- 5 - error

(default: 3)

EDN= - Alternate error listing dataset (default: no dataset)

ON= - Options to be enabled (default: C E L P Q R S T V)

OFF= - Options to be disabled (default: A B D F G H I J N O W X Z)

- A - abort if errors
- B - list sequence number of code generation block
- C - list common block names and lengths
- D - list DO-loop table
- E - recognize compiler directives
- F - FLOWTRACE
- G - list generated code (use only if requested by User Services)
- H - list only first statement of each program unit
- I - generate label symbol table
- J - one-trip DO-loops
- L - recognize listing control statements
- M - ignored
- N - put null symbols in symbol table
- O - identify out-of-bound array references
- P - allows double precision
- Q - abort on 100 fatal errors
- R - round multiply results
- S - list source code
- T - list symbol table
- V - vectorize inner DO-loops
- W - do not use
- X - include crcss-reference

```
                      Y - ignored
                      Z - put DEBUG symbol table on $BLD

TRUNC=         - number of bits to be truncated
                 (default: 0; maximum: 47)

AIDS=          - number of vectorization inhibition
                 messages
                  LOOPNONE - no messages
                  LOOPPART - maximum of 3 per inner
                             loop; 100 per compilation
                  LOOPALL  - all messages
                 (default: LOOPPART)

OPT=           - options (no more than one from each
                 of the following groups;
                 OPT=opt:opt:...):
                 . constant increment integer
                   optimization:
                    NOZEROINC - no incrementation by
                                zero value variables
                    ZEROINC   - incrementation by
                                zero value variables
                   (default: NOZEROINC)

                 . optimization for 1-line DO-loop
                   replacement with $SCILIB call:
                    SAFEDOREP - no replacement if
                                DO-loop has potential
                                dependencies or
                                equivalenced variables
                    FULLDOREP - alway replace
                    NODOREP   - never replace
                   (default: SAFEDOREP)

                 . move invariant code outside of
                   DO-loop:
                    INVMOV   - enable
                    NOINVMOV - disable
                      (default: INVMOV)

                 . instructions moving over a branch
                   instruction:
                    UNSAFEIF - enable
                    SAFEIF   - disable
                   (default: SAFEIF)

                 . bottom loading of scalar loops:
                    BL   - enable
                    NOBL - disable
                   (default: BL)
                  . B and T register allocation:
                     BTREG    - allocate maximum of 24
                                scalars to T regs
                     NOBTREG - allocate to memory
                    (default: NOBTREG)

                  . compilation of loops with specific
                    ambiguous dependencies in vector
                    and scalar versions:
                     CVL   - enable
                     NOCVL - disable
                    (default: enabled)

                  . update scalar temporaries in
                    DO-loops:
                     KEEPTEMP - enable
                     KILLTEMP - disable
                    (default: enable)

MAXBLOCK=   - number of words in a block of code to
              optimize or vectorize
MAXBLOCK=1 - disable
              (default: 2310; MAXBLOCK=1: disable)

INT=        - integer lengths
               64 - full 64-bit integers
               24 - short 24-bit integers
              (default: 64)

ALLOC=      - static memory allocation
               STATIC - all memory
               STACK  - read-only constants and
                        DATA, SAVE and common
                        block entities
               HEAP   - deferred implementation
              (default: STATIC)

CPU=        - mainframe type and hardware charac-
              teristics for running generated code
              cpu type:
               CRAY-XMP - 1, 2 or 4 processors
               CRAY-X1  - single-processor
               CRAY-X2  - dual-processor
               CRAY-X4  - quad-processor
              (default: compiling machine)

             hardware characteristics:
               [NO]EMA  - extended memory
               [NO]CI   - compressed index
               [NO]GS   - gather/scatter
               [NO]CIGS - compressed index gather/
                          scatter
               [NO]VPOP - vector popcount
                          functional unit
                    [NO]AVL  - two vector logical
                               functional units
                    [NO]BDM  - bidirectional memory

UNROLL=     - iteration count for unrolling inner
              DO-loops
              (range: 0 <= r <= 9)
              (default: 3)
UNROLL=0    - turn off unrolling

LOOPMARK=   - draw DO-loop brackets in source
              listing
               MSGS   - reasons for not vectorizing
               NOMSGS - no messages
              (default: NOMSGS)
LOOPMARK    - same as LOOPMARK=NOMSGS

DEBUG       - put sequence number labels in Debug
              Symbol Table
              (forces ON=IW and MAXBLOCK=1)
              (default: debugging turned off)

SAVEALL     - allocate user variables to static
              storage; compiler-generated variables
              to B or T registers

ANSI        - flag non-ANSI usage
```

Remarks: CFT compiles faster than CFT77, but executes more slowly.

See also: CFT77

Similar commands:

NOS/BE, NOS:	FTN5
VMS:	FORTRAN

Examples:

```
CFT.
CFT,I=$CPL.     <-- from UPDATE
CFT,LOOPMARK=MSGS.
CFT,B=myobj.
```

```
CFT77   Compile a Fortran 77 source program.

        Syntax       CFT77,I=idn,L=ldn,B=bdn,C=cdn,E=m,OPT=option,
                           INTEGER=il,ALLOC=alloc,ON=string,
                           OFF=string,TRUNC=nn,CPU=cpu:hdw,DEBUG,
                           LIST,STANDARD.

        Parameters:  I L B C E ALLOC TRUNC CPU DEBUG - same as CFT

                     OPT=     - at most one from each of the following
                                groups (OPT=opt:opt):
                                . optimization:
                                   FULL    - attempt full optimization
                                   OFF     - no optimization
                                             (fast compile)
                                   NOVECT - scalar optimization only
                                  (default: FULL)

                                . constant increment integer
                                  optimization:
                                   NOZEROINC - no incrementation by
                                               zero-value variables
                                   ZEROINC   - incrementation by
                                               zero-value variables
                                  (default: NOZEROINC)

                     INTEGER= - integer length
                                 64 - full 64-bit integers
                                 46 - short 46-bit integers
                                (default: 46)

                     ON=      - (default: P Q R)

                     OFF=     - (default: A C F G H J O S X)

                     LIST     - full compilation listing (sets ON=CGSX)
                                DO NOT USE -- specify ON=CSX instead

                     STANDARD - flag non-standard Fortran 77 usage

        Remarks:     CFT77 compiles much more slowly than CFT, but
                     may execute faster.  OPT=OFF does not vectorize
                     and will, therefore, run slower.

        See also:    CFT

        Similar commands:  NOS/BE, NOS:  FTN5
                           VMS:          FORTRAN

        Examples:    CFT77.
                     CFT77,I=$CPL.      <-- from UPDATE
                     CFT77,B=myobj.
```

```
CHARGES  Report on job resources.

         Syntax:   CHARGES,SR=options.

         Options:  CPU  - CPU, I/O wait, and CPU wait times since
                          start of job
                   DS   - permanent dataset statistics
                   JNU  - job name and user number
                   MM   - job size statistics
                   NBF  - number of blocks received from/queued to a
                          front end
                   TASK - CPU, I/C wait, and CPU wait times broken
                          down by task; and totals for job
                   WT   - time spent waiting in input queue

         Remarks:  CHARGES is invoked automatically at job end.

         Similar commands:  NOS/BE:  SUMMARY; ASSETS
                            NOS:     ENQUIRE
                            VMS:     ^T

         Examples: CHARGES,SR=DS:MM:TASK


COMPARE  Compare two datasets.

         Syntax:      COMPARE,A=adn,B=bdn,L=ldn,DF=df,ME=maxe,CP=cpn,
                              CS=csn,{CW=cw|CW=cw1:cw2},ABORT=ac.

         Parameters:  A=     - input dataset names - error if adn=bdn
                      B=

                      L=     - name of dataset for list of differences
                                 (default: $OUT;
                                           may not be same as adn or bdn)

                      DF=    - input dataset format
                                   B - binary - datasets compared
                                                logically with difference
                                                listed in octal
                                   T - text   - differences printed as
                                                text
                                 (default: T)

                      ME=    - maximum number of differences to be
                               printed
                               (default: 100)

                      CP=    - amount of context printed, that is, the
                               number of records on either side of a
                               difference to be printed - applies only
                               to DF=T)
                               (default: 0)
```

CS= - amount of context to be scanned, that is, the number of records on either side of a discrepancy to be scanned - applies only to DF=T)
(default: 0)

CW= - compare width - either compare columns 1 through cw or columns cw1 through cw2
(default: CW=1:133)

ABORT= - abort the job step after ac or more differences have been found

ABORT - same as ABORT=1
(default: 1)

Similar commands:

NOS/BE:	COMPARE; COMPAR
NOS:	VERIFY; VFYLIB
VMS:	DIFFERENCES

Examples:

```
ACCESS,DN=one,PDN=myfile1.
ACCESS,DN=two,PDN=myfile2.
COMPARE,one,two.
```

COPYD Copy blocked datasets.

Syntax: `COPYD,I=idn,O=odn,S=m.`

Parameters: S=m - shift count (number of ASCII blanks to be inserted at the start of each line)
(maximum: 132)

S - same as S=1
(default: 0)

Similar commands:

NOS/BE:	COPY; COPYF (DTRC); COPYR (DTRC); COPYSBF; COPYSF (DTRC); COPYSR (DTRC)
NOS:	COPY; COPYSBF
VMS:	COPY

Examples:

```
COPYD,I=myprog,S=25.
                ^-- copy shifted file to $OUT
                    (source program centered on wide
                    paper)
```

COPYF Copy blocked files.

Syntax: `COPYF,I=idn,O=odn,NF=nf,S=m.`

Parameters: I O S - same as COPYD

NF=nf - decimal number of files to copy
(default: 1)

Remarks: After the copy, 'both datasets are positioned after the EOF for the last file copied. If BFI=OFF is specified on the ASSIGN, compressed blanks are expanded.

Similar commands: NOS/BE: COPYBF; COPYCF; COPYF (DTRC);
COPYSBF; COPYSF (DTRC)
NOS: COPY; COPYBF; COPYCF; COPYSBF
VMS: COPY

Examples: `COPYF,I=FT02.` <-- print Fortran unit 2 on $OUT.

COPYR Copy blocked records.

Syntax: `COPYR,I=idn,O=odn,NR=nr,S=m.`

Parameters: I O S - same as COPYD

NR=nr - decimal number of records to copy (default: 1)

Remarks: After the copy, both datasets are positioned at the end of the last record copied. If BFI=OFF is specified on the ASSIGN, compressed blanks are expanded.

Similar commands: NOS/BE: COPYRE; COPYS; COPYSEL (all DTRC)

Examples: `COPYR,I=myfile,O=recs,NR=342.`

COPYU Copy unblocked datasets.

Syntax: `COPYU,I=idn,O=odn,NS=ns.`

Parameters: I O - same as COPYD

NS=ns - number of sectors to copy (default: 1)
NS - copy through EOD

Examples: `COPYU,I=unfyl1,O=unfyl2,NS.`

&DATA Defines the beginning of data within a procedure.

Syntax: `&DATA,dn.`

Parameters: dn - the name of the dataset to contain the data which follows this statement

Remarks: All lines following an &DATA up to the next &DATA or ENDPROC are written to the specified dataset.

Similar commands:

NOS/BE, NOS:	.DATA
VMS:	OPEN,WRITE,CLOSE

Examples:

```
PROC,MYPROC.
  ...
ENDPROC.
&DATA,IN1.
1.73, 2.6, 4
4.62, 9.7, 6
0,0,0
&DATA,IN2.
06Test01
12Ship 472-396X
```

DEBUG Interpret a dump.

Syntax:

```
DEBUG,S=sdn,L=ldn,DUMP=ddn,CALLS=n,TASKS,
      SYMS=sym[:sym],NOTSYMS=nsym[:nsym],
      MAXDIM=dim,BLOCKS=blk[:blk],
      NOTBLKS=nblk[:nblk],RPTBLKS,PAGES=np.
```

Parameters:

S= - Debug symbolic tables
(default: $DEBUG)

L= - Listable output
(default: $OUT)

DUMP= - Dump dataset name
(default: $DUMP)

CALLS= - Number of routine levels to display
(default: 50)

TASKS - Trace back through all existing tasks
(default: only through tasks running when dump taken)

SYMS= - List of symbols to be displayed
(Maximum: 20 symbols)
(default: all symbols)

NOTSYMS= - List of symbols to be skipped
(Maximum: 20 symbols)
(default: all symbols displayed)

MAXDIM= - Maximum number of each dimension to be displayed
(default: 20:5:2:1:1:1:1)

BLOCKS= - List of common blocks to include
(Maximum: 20 symbols)

BLOCKS - Include all common blocks

```
                  NOTBLKS= - List of common blocks to exclude
                             (overrides BLOCKS)
                             (Maximum: 20 symbols)
                  NOTBLKS  - Exclude all but subprogram block

                  RPTBLKS  - Repeat blocks (display with each
                             subprogram
                             (default: display once)

                  PAGES=   - Page limit
                             (default: 70)

       Similar commands: NOS/BE, NOS:  FTN5,PMD
                         VMS:          FORTRAN/DEBUG


DELETE Remove a permanent dataset.

       Syntax:    DELETE,DN=dn,NA,ERR,MSG,PARTIAL.

                  DELETE,PDN=pdn,ID=uid,OWN=owner,ED=ed,M=mn,
                         NA,ERR,MSG.

       Parameters: PARTIAL - delete the contents of the file, but
                             not the information about the file

                  ED=ed    - edition number (1-4095)
                              unsigned - specific edition
                              +n       - delete n highest editions
                              -n       - keep   n highest editions
                              ALL      - all editions
                             (default: highest edition)

       Remarks:   The first form is used if the permanent file has
                  already been ACCESSed.

                  The second form does not ACCESS the file.

       Similar commands: NOS/BE:  ALTER; PURGE
                         VMS:     CREATE a new version, PURGE/KEEP=1;
                                  DELETE; PURGE

       Examples:  ACCESS,myfile,UQ.
                  DELETE,DN=myfile,PARTIAL.
                  = = = = =
                  DELETE,PDN=myfile,ALL.
                  = = = = =
                  DELETE,PDN=A**.
                              ^-- delete all datasets with
                                  3-character names starting with
                                  "A"
```

DISPOSE Stage a dataset to the front-end; release a local dataset; change disposition characteristics.

```
Syntax:      DISPOSE,DN=dn,SDN=sdn,DC=dc,MF=mf,SF=sf,ID=uid,
                     TID=tid,R=rd,W=wt,M=mn,TEXT='text',DF=df,
                     WAIT¦NOWAIT,DEFER,NRLS.

Parameters:  DN=dn    - required

             SDN=sdn  - staged dataset name (1-15 characters)
                        (default: dn; required for CYBER 860)

             DC=dc    - to 860: DC=ST is required
                        to VAX: DC=PR with TEXT='any' makes a
                                file with Fortran carriage
                                control; DC=ST (with TEXT='any')
                                makes a file with carriage
                                return carriage control

             SF=sf    - special forms (1-8 alphanumeric
                        characters)
                        (default: no special forms)

             DF=df    - TR or CB or BB
                        (default: CB)

             WAIT     - wait or don't wait until dataset has
             NOWAIT     been staged to the front-end
                        (default: NOWAIT)

             DEFER    - disposition occurs at end-of-job or
                        when the dataset is RELEASEd

             NRLS     - after disposition, the dataset remains
                        local (use WAIT)

See also:    MSSTORE

Similar commands:  NOS/BE:  BEGIN,COMQ (DTRC);
                            BEGIN,XEROX (DTRC); ROUTE
                   NOS:     ROUTE
                   VMS:     FICHE (DTRC); PRINT; XEROX (DTRC)

Examples:    DISPOSE,DN=out1,DC=PR.
                          ^-- to VAX (assumed job origin)
             - - - - -
             DISPOSE,DN=out2,SDN=mymss,MF=N1,DC=ST,^
                     TEXT='USER,user,pw.'^
                          'PURGE,mymss/NA.'^
                          'DEFINE,mymss.'^
                          'CTASK.',WAIT.
                          ^-- send to MSS
             - - - - -
```

```
DISPOSE,DN=out3,MF=V3,^
          TEXT='myvax.dat',WAIT.
                ^-- send to VAXcluster
= = = = =
DISPOSE,DN=DISPLOT,DC=ST,DF=BB,TEXT='plot.dat',^
          WAIT.
                ^-- DISSPLA output file to VAX for
                     post processing
```

DS List local datasets.

```
Syntax:          DS.
```

Remarks: The information displayed includes alis, size, position (e.g., EOF), last operation, and open status.

```
Similar commands:  NOS:      ENQUIRE,F
                   NOS/BE:   FILES

Examples:        DS.
```

DSDUMP Dump a dataset in octal or hexadecimal.

```
Syntax:          DSDUMP,I=idn,O=odn,DF=df,IW=n,NW=n,IR=n,NR=n,
                          IF=n,NF=n,IS=n,NS=n,Z,DB=db,DSZ=sz.

Parameters:      I=    - (synonym: DN=idn)

                 O=    - dataset to receive the dump
                         (default: $OUT)

                 DF=   - dataset format
                           B - blocked
                           U - unblocked
                         (default: B)

                 IW=   - decimal/octal number of the initial word
                         for each record/sector
                         (defaults: 0 (Z specified);
                                    1 (Z omitted))
                 NW=   - decimal/octal number of words to dump
                         (default: 1)
                 NW    - through end of record/sector

                 IR=   - decimal/octal number of the initial record
                         for each input file - only if DF=B
                         (defaults: 0 (Z specified);
                                    1 (Z omitted))
                 NR=   - decimal/octal number of records per file
                         to dump - only if DF=B
                         (default: 1)
                 NR    - all records in each file
```

```
                      IF=  - decimal/octal number of the initial file in
                             idn - only if DF=B
                             (defaults: 0 (Z specified);
                                        1 (Z omitted))
                      NF=  - decimal/octal number of files to dump -
                             only if DF=B
                             (default: 1)
                      NF=0 - all files in the dataset

                      IS=  - decimal/octal number of the initial
                             sector - only if DF=U
                             (defaults: 0 (Z specified);
                                        1 (Z omitted))
                      NS=  - decimal/octal number of sectors to dump -
                             only if DF=U
                             (default: 1)

                      Z    - the zero-base for the initial-value
                             parameters (IW, IR, IF, IS)
                              Z    - each Ix is relative to 0;
                                     output refers to word, record,
                                     file, and sector numbers start
                                     at 0
                                       DSDUMP,...,IW=4096.  is same as
                                       DSDUMP,...,Z,IW=4095.
                              no Z - each Ix is relative to 1
                             (does not affect Nx parameters)

                      DB=  - numeric base for displaying the data words
                              OCTAL or O - octal
                              HEX   or H - hexadecimal

                      DSZ= - size of data items to dump
                              WORD   or W - words   (64 bits)
                              PARCEL or P - parcels (16 bits)
                             (default: WORD)

          Similar commands:  NOS/BE:  PRUDUMP; TAPDMP9; TDUMP (all DTRC)
                             NOS:     TDUMP

          Examples:     DSDUMP,I=myfile,NW=25,NR=5,DB=H.
                                     ^-- hexadecimal dump of first 25
                                         words of first 5 records of
                                         MYFILE


DUMP      Display job information previously captured by DUMPJOB.

          Syntax:       DUMP,I=idn,O=odn,FWA=fwa,LWA=lwa,JTA,NXP,V,DSP,
                             FORMAT=f,CENTER.

          Parameters:   I=      - dataset containing the memory image
                                  (default: $DUMP)
```

FWA= - first word address to dump
(default: word 0 of Job Communication Block (JCB))

LWA= - last word address to dump
(default: 200 of JCB)

LWA - the limit address

LWA=0 - no memory

JTA - dump Job Table Area
(default: no JTA dump)

NXP - dump No Exchange Package, B, T, cluster, and semaphore registers
(default: these are dumped;
NXP overrides V if both specified)

V - dump vector registers
(default: do not dump vector registers)

DSP - dump Logical File Tables (LFTs) and Dataset Parameter Tables (DSPs)
(default: do not dump LFTs and DSPs)

FORMAT= - format for dumping FWA through LWA

```
D - data  - decimal integer and ASCII
G - data  - floating-point or
            exponential and ASCII
I - instr - CAL mnemonics and ASCII
M - data  - each 16-bit parcel
            displayed as 1 hex and 4
            octal digits
C - data  - octal integer and ASCII
P - data  - 16-bit parcel
X - data  - hex integer and ASCII
```

CENTER - dump 100 (octal) words on each side of P-register address in P format

Similar commands: NOS/BE, NOS: DMD, DMP

Examples: See DUMPJOB.

DUMPJOB Capture job information in dataset $DUMP for display by DUMP.

Syntax: DUMPJOB.

Examples:

```
...
EXIT.
DUMPJOB.
DUMP,....
DUMP,....
```

```
ECHO     Control logfile messages.

         Syntax:      ECHO,ON=class1:...:classm,OFF=class1:...:classn

         Parameters:  ON=  - list of classes whose messages are to be
                             written to the log file
                             ("ON" is the same as "ON=ALL")
                      OFF= - list of classes whose messages are NOT to
                             be written to the log file
                             ("OFF" is the same as "OFF=ALL")

                      classi - ABORT   - job failure
                               JCL     - messages in user's JCL
                               PDMERR  - PDM errors
                               PDMINF  - PDM dataset information
                               ALL     - all classes

         Remarks:     The ECHO state after returning from a procedure
                      call is the same as before the call, regardless
                      of any changes made in the procedure.

                      Within a procedure, the ECHO state is that of the
                      caller, unless changed within the procedure.

         Similar commands:  NOS/BE:  DAYFILE

         Examples:    ECHO,OFF.


ELSE     See IF.


ELSEIF   See IF.


ENDIF    See IF.


ENDLOOP  See LOOP.


ENDPROC  See PROC.
```

EXIT On job abort, processing continues with the statement following the EXIT; if no abort, terminate job processing.

Syntax: EXIT.

```
Similar commands:  NOS/BE, NOS:  EXIT
                   VMS:          ON condition
```

```
Examples: ...
          EXIT.
          DUMPJOB.
          DUMP.
          ...
```

EXITIF See IF.

EXITLOOP See LOOP.

FETCH Get a front-end dataset and make it local.

```
Syntax:       FETCH,DN=dn,SDN=sdn,AC=ac,TEXT='text',MF=mf,
                    DF=df.
```

```
Parameters:   DN=   - local dataset name

              SDN=  - staged dataset name (front-end dataset
                      name)
                      (default: dn)

              AC=   - acquisition code (where the dataset is to
                      be acquired)
                        IN - input (job) dataset - use SUBMIT
                             to run the job
                        IT - intertask communication
                        MT - magnetic tape at the front end
                        ST - staged dataset from the front end
                      (default: ST)

              MF=   - mainframe computer identifier
                        N1 - MSS
                        V3 - DT3
                      (default: front end of job origin)

              DF=   - dataset format (BB, BD, CB, CD, TR)
                      (default: CB)
```

Remarks: FETCH defaults to DF=CB, MSFETCH defaults to DF=TR.

See also: MSFETCH

```
        Similar commands:  NOS/BE:  MSFETCH (get an MSS file, DTRC)
                           VMS:     HFT FETCH (get an MSS file, DTRC)

        Examples:     FETCH,DN=SOURCE,TEXT='PROG.FOR'.
                      = = = = =
                      FETCH,DN=FT11,DF=TR,^
                            TEXT='[ABCD.SUBD1]CRAYBIN.DAT'.
                                   ^-- binary data file from a VAX
                                       subdirectory of user ABCD
                      = = = = =
                      FETCH,DN=SORC,SDN=mssname,MF=N1,^
                            TEXT='USER,name,pw.^
                                 'GET,mssname.CTASK.'.
                                   ^-- get an indirect MSS (860) file


FLODUMP Dump flowtrace table of a program abort.

        Syntax:       FLODUMP,L=ldn.

        Parameters:   L= - dataset to contain the report
                           (default: $OUT)

        Examples:     ...
                      EXIT.
                      DUMPJOB.
                      FLODUMP.


FTREF   Generate Fortran cross-reference.

        Syntax:       FTREF,I=idn,L=ldn,CB=op,TREE=op,ROOT=root,
                            END=end,LEVL=n,DIR=dir,NORDER,MULTI.

        Parameters:   I=     - input dataset containing the cross-
                               reference table listing and Fortran
                               source program (ON=XS)

                      CB=    - global common block cross references
                                 PART - routines using a common block
                                 FULL - use of common block variables
                                 NONE - no output information
                               (default: PART)

                      TREE=  - static calling tree
                                 PART - entry names, external calls,
                                        calling routines, common block
                                        names
                                 FULL - PART plus static calling tree
                                 NONE - no output information
                               (default: PART)
```

ROOT= - if TREE=FULL, this defines the name of the routine to be used as the root of the tree
(default: the routine not called by any other routine;
if more than one, the first alphabetically)

END= - if TREE=FULL, this defines the name of the routine to be used as the end of any branch of a tree
(default: complete trees are generated)

LEVEL= - if TREE=FULL, this is the maximum length of any branch
(default: the entire program)

DIR= - dataset containing processing directives
(default: no directives)

NORDER - list subprograms in input order
(default: list in alphabetical order)

MULTI - summarize multitasking subroutine usage

Directives: The following may be in the DIR= dataset:

ROOT - list of modules to be used as roots of separate trees
ROOT,md1,md2,...,mdn.

SUBSET - list of modules to be processed
SUBSET,md1,md2,...,mdn.
(default: all modules)

CHKBLK - list of common blocks to be checked for locked variables
CHKBLK,blk1,blk2,...,blkn.

CHKMOD - list of external calls to be checked for calling from a locked area
CHKMOD,mod1,mod2,...,modn.

Similar commands: NOS/BE, NOS: FTN5,LO=
VMS: FORTRAN /CRCSS_REFERENCE

HOLD Specify that dataset release occurs with implicit HOLD.

Syntax: HOLD,GRN=grn.

Parameters: GRN=grn - generic resource name

Remarks: This prevents return of resources to the system and is useful when dataset assignment is done by applications over which the user has no control.

See also: NOHOLD

IF Begin a conditional block of code.

Syntax:

```
IF(expression)
  <do if true>
ELSEIF(expression)
  <do if true>
ELSE.
  <do if all other tests fail>
ENDIF.

EXITIF.                   <-- exit unconditionally
EXITIF(expression)        <-- exit if exp is true
```

Parameters: exp - a valid JCL expression

Remarks: Literal strings, '...', in an IF/ELSEIF expression are limited to 8 characters (one machine word).

Similar commands:

NOS/BE:	IFE
NOS:	IF; IFE
VMS:	IF

Examples:

```
ACCESS,DN=MYPROG,NA.
IF(PDMST.NE.1)
  UPDATE(Q=MYPROG)
  CFT(I=$CPL,ON=A)
  NOTE(DN=SLIN,TEXT='ABS=MYPROG')
                    ^-- create input directive
                        file for SEGLDR
  SEGLDR(I=SLIN)
  SAVE(DN=MYPROG,NA)
  EXITIF.
  EXIT.
*.
*.     Error while generating MYPROG
*.
  EXIT.
ENDIF.
MYPROG.
====================
```

Same as above, but in a procedure, with SEGLDR directives in a data file in the procedure:

```
PROC.
DOMYPROG.
   ...        <-- omit NOTE command
ENDPROC.
&DATA,SLIN
   ...
ABS=MYPROG
```

IOAREA Control access to a job's I/O area (containing the DSP and I/O buffers).

Syntax: IOAREA, { LOCK ¦ UNLOCK }

Parameters: LOCK - the limit address is set to the base of the DSPs, denying direct access to the user's DSP and I/O buffers. When locked, system I/O routines can gain access.

UNLOCK - the limit address is set to JCFL, allowing access to these areas.

Examples: IOAREA,LOCK.

ITEMIZE Report statistics about a library dataset.

Syntax: ITEMIZE,DN=dn,L=ldn,NREW,NF=n,T,BL,E,B,X.

Parameters: DN= - (default: $OBL)

NREW - no rewind
(default: rewind before and after)

NF= - number of files to be listed
(default: 1)
NF - all files

T - truncate lines after 80 characters
(if specified, E, B, X may not be used)

BL - burstable listing (each heading is at top of a page
(default: page eject only when current page is nearly full)

E - list all entry points (binary library datasets only)

B - E plus code and common block information
(B overrides E)

```
                  X    - B plus external information
                           (X overrides B)

             Restrictions:  . an UPDATE PL is recognized only if it is the
                              only item in a dataset
                            . standard COS blocked datasets only

             Similar commands:  NOS/BE:  ITEMIZE; LISTBIN (DTRC)
                                NOS:     ITEMIZE
                                VMS:     LIBRARIAN

             Examples:      ITEMIZE,DN=myreloc
                            ITEMIZE,DC=mylib,X.


JOB          First statement of a job - gives job parameters.

             Syntax:        JOB,JN=jn,MFL=fl,T=tl,OLM=olm,US=jcn.

             Parameters:    JN=jn    - job name (1-7 alphanumeric characters)

                            MFL=fl   - maximum field length (decimal) for the
                                       job - fl is rounded up to the nearest
                                       multiple of 512 words, or the amount
                                       needed to load CSP (Control Statement
                                       Processor)
                                       (default: 512000)
                            MFL      - the system maximum (3,536,000)

                            T=tl     - job time limit (decimal seconds)
                                       (default: 30; max: 200000)
                            T        - the system maximum (~194 days!)
                                       NOTE: your job will not run because
                                             this exceeds the DTRC maximum!

                            OLM=olm  - maximum size of $OUT; olm is the
                                       number of 512-word blocks (each block
                                       holds about 45 lines)
                                       (default: 2000; maximum: 8192)

                            US=jcn   - job class (1-7 alphanumeric characters)
                                       jcn is one of:
                                         EXPRESS, NORMAL, DEFER, SECURE
                                       Job is dropped to a lower class if it
                                       doesn't fit the requested job class.
                                       (default: NORMAL, if it fits)
                                       (see page 2-1-3 for the job class
                                       limits)

             See also:      ACCOUNT

             Similar commands:  NOS/BE, NOS:  job statement
```

Examples:

```
JOB,JN=jobname1.
ACCOUNT,....
<rest of job>
```

JOBCOST (DTRC - UTILITY) Write a summary of the job cost and system usage to $LOG.

Syntax: `JOBCOST`

Remarks: A subroutine version is available in DTLIB.

Similar commands: NOS/BE: SUMMARY

Examples:

```
ACCESS,DN=UTILITY,OWN=PUBLIC.
LIBRARY,DN=UTILITY:*.
JOBCOST.      <-- the cost to this point in job
< execute your program >
JOBCOST.      <-- the cost of running your program
```

LIBRARY Specify the library dataset search order for control statement verbs.

Syntax: `LIBRARY,DN=dn1:dn2:...:dn64,V.`

Parameters:

```
DN= - up to 64 library names to be searched - an
      asterisk means add the listed names to the
      current searchlist
V   - list the current library searchlist in the
      logfile
```

Similar commands: NOS/BE, NOS: LIBRARY; LDSET,LIB= (not subs)

Examples:

```
LIBRARY,DN=THISLIB:YOURLIB.
                            ^-- the searchlist contains
                                2 libraries
LIBRARY,DN=THATLIB:*,V.
                            ^-- the searchlist now has
                                3 libraries and are
                                listed in the logfile
LIBRARY,,V.                 <-- list the current
                                searchlist in the logfile
```

LOOP Start of an iterative control statement block.

Syntax:

```
LOOP.
...
EXITLOOP.
EXITLOOP(expression)
...
ENDLOOP.
```

Parameters: exp - a valid JCL expression

```
Similar commands:  NOS/BE, NOS:  WHILE

Examples:     Merge two datasets for 60 records:
                SET,J1=0.
                SET,J2=60.
                LOOP.
                  EXITLOOP(J2.EQ.0)
                  IF(J1.EQ.0)
                    COPYR,I=DSIN1,O=OUTDS.
                    SET,J1=1.
                  ELSE.
                    COPYR,I=DSIN2,O=OUTDS.
                    SET,J1=0.
                  ENDIF.
                  SET,J2=J2-1.
                ENDLOOP.
                REWIND,DN=DSIN1:DSIN2:OUTDS.
```

MEMORY Request new field length.

```
Syntax:       MEMORY.
              MEMORY,FL=fl.
              MEMORY,FL=fl,{ USER | AUTO }.

Parameters:   FL=fl  - the decimal number of words of field
                       length; "FL" allocates the job maximum
              USER   - field length is retained until the next
                       request
              AUTO   - field length is reduced automatically at
                       the end of each job step

Similar commands:  NOS/BE:  RFL
                   NOS:     MFL

Examples:     MEMORY,FL,USER.         <-- get and hold the maximum
                                          field length
              MEMORY,AUTO.            <-- resume automatic mode
                                          (FL reduces after next job
                                          step)
              MEMORY,FL=32978.        <-- get and hold 32978 words
                                          (user mode)
              MEMORY,FL=32978,AUTO.
                                      ^-- get 32978 words for next
                                          job step only
```

MODE Set/clear mode flags.

```
Syntax:       MODE,FI=option,BT=option,EMA=option,AVL=option,
                   ORI=option.

Parameters:   option - ENABLE or DISABLE
              FI     - floating-point error interrupts
                       (default: ENABLE)
```

BT - bidirectional memory transfers
(default: ENABLE)
EMA - extended memory addressing
(default: DISABLE)
AVL - second vector logical function unit
(default: DISABLE)
ORI - operand range error interrupt
(default: ENABLE)

Similar commands: NOS/BE, NOS: MODE
VMS: ON condition

MODIFY Change a permanent dataset's characteristics.

Syntax:

```
MODIFY,DN=dn,PDN=pdn,ID=uid,ED=ed,RT=rt,R=rd,
    W=wt,M=mn,NA,ERR,MSG,EXO=exo,PAM=mode,ACN.
```

Parameters: RT=rt - new retention period
RT= - reset to default

ACN - use the alternate account number

See also: ALTACN, SAVE

Similar commands: NOS/BE: RENAME
NOS: CHANGE
VMS: SET PROTECTION

Examples:

```
ACCESS,DN=mylocal,PDN=myperm,UQ,M=maint.
MODIFY,DN=mylocal,PAM=R.
```

MSACCES (DTRC - PROCLIB) Supply your Username and password to the Mass Storage System.

Syntax:

```
MSACCES,US=us,MPW=mpw.
```

Parameters: UN=us - your Username (User Initials)

MPW=mpw - your MSS password

Remarks: MSACCES is required before using the MSx commands.

Similar commands: NOS/BE: MSACCES (DTRC)
VMS: HFT ACCESS (DTRC)

Examples:

```
ACCESS,DN=PROCLIB,OWN=PUBLIC.
LIBRARY,DN=PROCLIB:*.
MSACCES,UN=myid,MPW=mymsspw.
MSFETCH,.... -or- MSPURGE,.... -or- MSSTORE,....
```

MSFETCH (DTRC - PROCLIB) Fetch a file from the Mass Storage System.

Syntax: `MSFETCH,DN=dn,MDN=mdn,DF=df,UN=un,PW=pw.`

Parameters:

DN=dn - the local dataset name

MDN=mdn - the MSS dataset (file) name
(default: MDN=dn)

DF=df - data format
TR - transparent (no conversion)
CB - character blocked (convert from CDC display code)
(default: DF=TR)

UN=un - Username (User Initials) of the owner of the MSS file
(omit for your own files)

PW=pw - optional MSS file password

Remarks: MSACCES is required before using the MSx commands.

MSFETCH defaults to DF=TR, FETCH defaults to DF=CB.

See also: ACQUIRE, FETCH

Similar commands:	
NOS/BE:	MSFETCH (DTRC)
NOS:	ATTACH
VMS:	HFT FETCH (DTRC)

Examples:

```
ACCESS,DN=PROCLIB,OWN=PUBLIC.
LIBRARY,DN=PROCLIB:*.
MSACCES,UN=myid,MPW=mymsspw.
MSFETCH,DN=in1,MDN=mymsfyl.
MSFETCH,DN=in2,MDN=hisfyl,UN=him,DF=CB,PW=fylepw.
```

IN1 is your file MYMSFYL transferred without conversion.

IN2 is file HISFYL belonging to user HIM converted from CDC Display Code (FYLEPW is the password HIM requires for access to the file).

MSPURGE (DTRC - PROCLIB) Purge a file from the Mass Storage System.

Syntax: `MSPURGE,MDN=mdn.`

Parameters: MDN=mdn - the MSS dataset (file) name
(default: MDN=dn)

Remarks: MSACCES is required before using the MSx commands.

```
Similar commands:  NOS/BE:  MSPURGE (DTRC)
                   NOS:     PURGE
                   VMS:     HFT DELETE; MSSDELETE (both DTRC)

Examples:     ACCESS,DN=PROCLIB,OWN=PUBLIC.
              LIBRARY,DN=PROCLIB:*.
              MSACCES,UN=myid,MPW=mymsspw.
              MSPURGE,MDN=mssfyl1.
```

MSSTORE (DTRC - PROCLIB) Store a file on the Mass Storage System.

```
Syntax:       MSSTORE,DN=dn,MDN=mdn,DF=df,CT=ct,NA=na,PW=pw.

Parameters:   DN=dn   - the local dataset name

              MDN=mdn - the MSS dataset (file) name
                        (default: MDN=dn)

              DF=df   - data format
                          TR - transparent (no conversion)
                          CB - character blocked (convert from
                               CDC display code)
                        (default: DF=TR)

              CT=ct   - Category
                          P  - private
                          PU - public
                          S  - semi-private
                        (default: CT=P)

              NA=na   - No Abort
                          0 - abort if file already exists on
                              the MSS
                          1 - replace the olf MSS file, is one
                              exists
                        (default: NA=0)

              PW=pw   - optional MSS file password

Remarks:      MSACCES is required before using the MSx commands.

See also:     DISPOSE

Similar commands:  NOS/BE:  MSSTORE (DTRC)
                   NOS:     DEFINE
                   VMS:     HFT STORE (DTRC)

Examples:     ACCESS,DN=PROCLIB,OWN=PUBLIC.
              LIBRARY,DN=PROCLIB:*.
              MSACCES,UN=myid,MPW=mymsspw.
              MSSTORE,DN=in1,MDN=mssfyl1.
              MSSTORE,DN=in2,MDN=mssfyl2,DF=CB,NA=1,PW=fylepw.

              IN1 is stored as private file MSSFYL1.
```

IN2 is stored as private file MSSFYL2 (even is MSSFYL2 already exists) in CDC Display Code. FYLEPW is the password required for another user to access the file.

NEWCHRG (DTRC - PROCLIB) Change permanent file account number.

Syntax: NEWCHRG,OLD=oldchrgno,ID=id.

Parameters:

```
OLD=   - the account number to be changed

ID=id  - change all files having this ID
ID     - change all files having a null ID
           (default: change all IDs)
```

Remarks: NEWCHRG changes from the specified account number to the "current" number of the Cray job (from the ACCOUNT or most recent ALTACN statement).

See also: ATLACN.

Similar commands:

```
NOS:     BEGIN,NEWCHRG
NOS/BE:  BEGIN,RENAMAC
```

Examples:

```
JOB,JN=....
ACCOUNT,AC=....
ACCESS,DN=PROCLIB,OWN=PUBLIC.
LIBRARY,DN=PROCLIB:*.
NEWCHRG,OLD=1222233344.
                ^-- change all files from account
                    1-2222-333-44 to the current one
= = = = =
...
NEWCHRG,OLD=1222233344,ID=myid.
                ^-- change all files WITH ID=MYID
                    from account 1-2222-333-44 to
                    the current one
= = = = =
...
ALTACN,AC=5666677788.
NEWCHRG,OLD=12222433344.
                ^-- change all files from account
                    1-2222-333-44 to 5-6666-777-88
```

NOHOLD Cancel the effect of HOLD.

Syntax: NOHOLD,GRN=grn.

Parameters: GRN=grn - generic resource name

See also: HOLD

```
NORERUN  Control a job's rerunability.

         Syntax:      NORERUN,option.

         Parameters:  option - ENABLE  - declare a job nonrerunable if
                                         any of the nonrerunable
                                         functions are done
                                DISABLE - stop monitoring nonrerunable
                                         functions (if a job has
                                         already been declared
                                         nonrerunable, that status is
                                         not changed)
                               (default: ENABLE)

         See also:    RERUN

         Similar commands:  NOS/BE:  NORERUN (DTRC)
                            NOS:     NORERUN

         Examples:    NORERUN,DISABLE.


NOTE     Write text to a dataset.

         Syntax:      NOTE,DN=dn,TEXT='text'.

         Parameters:  DN=    - the dataset to be written (at its current
                               position)
                      DN     - write to $OUT

                      TEXT= - up to 153 character to be written

         Similar commands:  NOS/BE:  NOTE (DTRC)
                            NOS:     NOTE
                            VMS:     OPEN,WRITE,CLOSE

         Examples:    NOTE,DN=UIN,TEXT='*COMPILE myprog,mysub'.
                      REWIND,UIN.
                      UPDATE,I=UIN,....


OPTION   Specify user-defined options.

         Syntax:      OPTION,LPP=n,PN={ p ¦ ANY },STAT=stat.

         Parameters:  LPP=n -   number of lines per page for job
                                listings (0-255 decimal)
                      LPP=0 -   do not change the current setting
                                (default: 66)

                      PN=p    - select a processor (p is 1 or 2)
                      PN=ANY  - any available processor
                                (if invalid, job aborts with an error
                                 message)
                                (default: ANY)
```

```
                STAT=   - the level of I/O statistics gathered
                          for local datasets to appear in the
                          user logfile
                          (user level   - accounting information
                           system level - device information)
                           ON    - installation defined
                           OFF   - no statistics
                           USR   - user information
                           FULL  - user and system info
                          (default: OFF)
                STAT    - same as STAT=ON
```

Similar commands: VMS: SUBMIT /QUEUE=

PASCAL Compile a Pascal source program.

```
Syntax:       PASCAL,I=idn,L=ldn,B=bdn,O=list,
                     CPU=cpu:char.

Parameters: B=   - generated binary load modules
                   (default: $BLD)

            O=   - Compiler options, separated by colons
                   (default: A-:BP-:BREG=8:BT-:C-:D+:H2:^
                             H+24:L+:O+:P-:R+:RV-:S4:S+4:^
                             ST-:T+:TREG=8:U-:V+:X-:Z+)

            CPU= - Cray to execute the program
                     cpu  - CRAY-XMP
                            CRAY-X1  - single-processor
                            CRAY-X2  - dual-processor
                            (default: the compiling machine)

                     char - [NO]EMA     - extended memory
                                          (24-bit A-register
                                          immediate loads;
                                          common blocks > 4
                                          million words)
                            [NO]CIGS    - compressed index
                                          scatter/gather
                            [NO]VPOP    - vector population
                                          and parity
                            [NO]READVL  - vector length read
                                          instructions
                            MEMSIZE=nK  - (n * 1024) words
                            MEMSIZE=nM  - (n * 1048576) words
                            [NO]BDM     - bidirectional memory
```

Similar commands: NOS/BE, NOS, VMS: PASCAL

Examples: PASCAL,I=mypasc.

PERMIT Grant/deny access to a permanent dataset.

Syntax:

```
PERMIT,PDN=pdn,ID=uid,AM=am,RP,USER=ov,ADN=adn,
       NA,ERR,MSG.
```

Parameters:

```
PDN=pdn - required

RP      - remove the permissions

USER=ov - the name (User Initials) of the user to
          be granted/denied permission

ADN=adn  - local dataset with the permit list
```

Similar commands:

```
NOS/BE:  MSCHANG; MSPERMT (both DTRC)
NOS:     CHANGE; PERMIT
VMS:     SET PROTECTION; Access Control List
```

Examples:

```
PERMIT,PDN=myfile,USER=abcd,AM=R.
            ^-- make file readonly for user ABCD
- - - - -
PERMIT,PDN=myfile,USER=abcd,AM=N.
            ^-- remove all permissions for user
                ABCD
```

PRINT Write the value of a JCL expression to the logfile.

Syntax:

```
PRINT(expression)
```

Parameters:

```
exp - any valid JCL expression
      (maximum length: 8 characters)
```

Logfile format: UT060 decimal octal ASCII

Similar commands:

```
NOS/BE, NOS:  DISPLAY
VMS:          WRITE SYS$OUTPUT
```

Examples:

```
SET(J1=J1+1)
PRINT,J1.
```

PROC Begin an in-line procedure definition block. This is followed by the procedure prototype statement which names the procedure and gives the formal parameter specifications.

Syntax:

```
PROC.
name,p1,p2,...,pn
   ...
ENDPROC.
```

Parameters:

```
name - the name of the procedure (1-8 alpha-
       numeric characters; should not be the same
       as a system verb)
```

```
pi   - a formal parameter specification in one
       of the following formats:

         pos               - positional
         key=dvalue:kvalue - keyword
           key    - formal keyword parameter
           dvalue - optional default value if
                    the parameter is omitted
           kvalue - optional value if the
                    parameter is specified with
                    no value
       special cases:
         key=  - specify a null value
         key=: - no defaults, but caller may
                 specify key= or just key
```

See also: Section 2-3

Similar commands:		
	NOS/BE, NOS:	.PROC
	VMS:	always 8 parameters

Examples:

```
PROC.
...
ENDPROC.
```

QUERY Determine the current status and position of a local file.

Syntax: QUERY,DN=dn,STATUS=status,POS=pos.

Parameters: STATUS= - the JCL symbol name to receive the status of the dataset - return values:

value	meaning
-1	dn is not local
0	dn is closed
1	dn is open for output
2	dn is open for input
3	dn is open for I/O

POS= - the JCL symbol name to receive the position of the dataset - return values:

value	meaning
-1	position indeterminate (not local, unblocked, closed)
0	BOD (beginning-of-data)
1	EOD (end-of-data)
2	EOF (end-of-file)
3	EOR (end-of-record)
4	mid-record

```
        Remarks:     In addition, a logfile message is generated:

                        QU001 - DN: ldn STATUS:  status  POS:  pos

                     where status is UNKNOWN, CLOSED, OPEN-O, OPEN-I,
                                     OPEN-I/O
                           pos    is N/A, BOD, EOD, EOF, EOR, MID

        Similar commands:  NOS/BE:  FILES
                           NOS:     ENQUIRE
                           VMS:     no local file concept

        Examples:    QUERY,DN=myfile,STATUS=stat,POS=pos.
                     IF(STATUS.LT.0)
                        COMMENT. file myfile is not local
                        ...
                     ELSE.
                        COMMENT. file myfile is local
                        ...
                     ENDIF.


RELEASE Return a dataset.

        Syntax:      RELEASE,DN=dn1:dn2:...:dn8,HOLD.

        Parameters:  DN=  - up to 8 dataset names
                     HOLD - hold generic resource (do not return the
                            allocation to the system pool)

        See also:    HOLD, NOHOLD

        Similar commands:  NOS/BE, NOS:  RETURN

        Examples:    RELEASE,DN=temp:filel:out.


RERUN   Control a job's rerunability.

        Syntax:      RERUN,option.

        Parameters:  option - ENABLE  - mark job as rerunable
                                        regardless of any
                                        nonrerunable functions
                                        which may have been
                                        performed so far in the job
                              DISABLE - mark the job as nonrerunable
                              (default: ENABLE)

        See also:    NORERUN

        Similar commands:  NOS/BE:  NORERUN (DTRC)
                           NOS:     NORERUN

        Examples:    RERUN,ENABLE.
```

```
RETURN   Return control from a procedure to its CALLer.

         Syntax:      RETURN.
                      RETURN,ABORT.

         Parameters:  ABORT - cause COS to issue a job step abort

         Similar commands:  NOS/BE, NOS:  REVERT
                            VMS:          EXIT

         Examples:    See PROC.


REWIND   Position a dataset at its beginning.

         Syntax:      REWIND,DN=dn1:dn2:...:dn8.

         Parameters:  DN= - up to 8 datasets to be rewound

         Similar commands:  NOS/BE, NOS:  REWIND

         Examples:    REWIND,DN=temp:out:in1.


ROLLJOB  Protect a job by writing it to disk.

         Syntax:      ROLLJOB.

         Remarks:     There is no guarantee that a job will remain
                      recoverable.

         Examples:    ROLLJOB.


SAVE     Make a local dataset permanent and define its characteristics.

         Syntax:      SAVE,DN=dn,PDN=pdn,ID=uid,ED=ed,RT=rt,R=rd,W=wt,
                           M=mn,UQ,NA,ERR,MSG,EXO=exo,PAM=mode,
                           ADN=adn,ACN.

         Parameters:  RT=rt    - retention period
                      RT=      - set to default

                      ADN=adn  - local dataset with the permit list

                      ACN      - use the alternate account number

         See also:    ALTACN, MODIFY

         Similar commands:  NOS/BE:  CATALOG
                            NOS:     DEFINE; SAVE
                            VMS:     CREATE
```

```
        Examples:     SAVE,DN=out,PDN=ABCOUT.
                      = = = = =
                      SAVE,DN=prog,PDN=mastprog,M=maint,PAM=R.
                                    ^-- the file is world-readable and
                                        YOU can't accidentally delete it
```

SCRUBDS Write over a dataset before release.

```
        Syntax:       SCRUBDS,DN=lfn.

        Parameters:   lfn - the uniquely accessed file to be
                            overwritten

        Remarks:      SCRUBDS writes zeros over an existing dataset.

        Examples:     ACCESS,DN=myfyl,PDN=myfyle,UQ.
                      SCRUBDS,DN=myfyl.
```

SEGLDR Segment loader.

```
        Syntax:       SEGLDR,I=idn,L=ldn,DW=dw,CMD='directives',GO.

        Parameters:   I=   - Dataset with SEGLDR directives
                             (default: $IN)
                      I    - Same as I=$IN

                      L=   - Listable output
                             (default: $OUT)
                      L    - Same as L=$OUT

                      DW=  - Input directive data width
                      DW   - Same as DW=80
                             (default: 80)

                      CMD= - Global directives to be processed;
                             treated as first record read from I=idn;
                             separate commands with semicolons
                             (e.g., CMD='BIN=bdn;MAP=PART')

                      GO   - Load and execute;
                             ignored for a segmented load

        Remarks:    By default, input load modules are read from $BLD.

        Directives:   See section 2-6.

        Similar commands:  NOS/BE, NOS:  SEGLOAD
                           VMS:          virtual machine

        Examples:     CFT,B=myobj.
                      SEGLDR,CMD='BIN=myobj;MAP=PART',GO.
```

```
SET     Change the value of a JCL variable.

        Syntax:      SET(symbol=expression)

        Parameters:  exp - a valid arithmetic, logical or literal
                           assignment expression - may be delimited by
                           parentheses

        Remarks:    The job-step aborts if the variable is unknown,
                     is changable only by COS, or is a constant.

        Similar commands:  NOS/BE, NOS:  SET
                           VMS:          $ name = value

        Examples: SET(J1=J1+1)     <-- increment procedure-local
                                       register J1 by 1

                  SET(G1=(SYSID.AND.177777B))
                                  ^-- put the low-order 2 characters
                                      of the current system revision
                                      level into global register G1

                  SET(G3=((ABTCODE.EQ.74).AND.(G2.EQ.0)))
                                  ^-- define global register G3


SID     Debug programs interactively or in batch.

        Syntax:       SID=adn,I=idn,S=sdn,L=lsn,ECH=edn,CNT=n.

        Parameters:  adn  - absolute dataset name (from LDR,AB=adn)

                     I=   - Input directives
                            (default: $IN)

                     S=   - Symbol dataset name
                            (default: $DEBUG)

                     L=   - Listable output
                            (default: $OUT)

                     ECH= - Dataset for echoing input directives
                            (default: no echoing)
                     ECH  - Same as ECH=ldn

                     CNT= - Breakpoint interrupt count
                            (default: 0 (no abort))

        Similar commands:  NOS/BE, NOS:  CID
                           VMS:          DEBUG
```

```
SKIPD    Skip blocked datasets (position at EOD (after last EOF)).

         Syntax:      SKIPD,DN=dn.

         Parameters:  DN - (default: $IN)

         Same as:     SKIPF,DN=dn,NF.

         Similar commands:  NOS/BE:  EOI (DTRC)
                            NOS:     SKIPEI
                            VMS:     OPEN with ACCESS=APPEND in program

         Examples:    SKIPD,DN=myfile.


SKIPF    Skip blocked files from current position.

         Syntax:      SKIPF,DN=dn,NF=nf.

         Parameters:  DN=dn  - (default: $IN)

                      NF=nf  - decimal number of files to skip forward
                      NF=-nf - decimal number of files to skip backward
                      NF     - position after the last EOF of the
                               dataset
                               (default: NF=1)

         Similar commands:  NOS/BE:  SKIPF; SKIPR (both DTRC)
                            NOS:     SKIPF; SKIPFB; SKIPR

         Examples:    SKIPF,DN=myfile.


SKIPR    Skip blocked records from the current position.

         Syntax:      SKIPR,DN=dn,NR=nr.

         Parameters:  DN=dn  - (default: $IN)

                      NR=nr  - decimal number of records to skip
                               forward
                      NR=-nr - decimal number of records to skip
                               backward
                      NR     - position after the last EOF of the
                               current file
                               (default: NR=1)

         Similar commands:  NOS/BE:  COPYS (DTRC)

         Examples:    SKIPR,DN=myfile.
```

```
SKIPU   Skip sectors on unblocked datasets.

        Syntax:      SKIPU,DN=dn,NS=ns.

        Parameters:  DN=dn   - no default

                     NS=ns   - decimal number of sectors to skip
                               forward
                     NS=-ns  - decimal number of sectors to skip
                               backward
                     NS      - position after the last sector of the
                               dataset
                               (default: NS=1)

        Examples:    SKIPU,DN=myfile.


SORT    Sort/merge.

        Syntax:       SORT,S=sdn[:sdn...],M=mdn[:mdn...],O=odn,
                           DIR=ddn,L=ldn,ECHO,RETAIN,NOVERF.

        Parameters:  S=      - Input dataset of up to 8 unsorted files
                     M=      - Input dataset of up to 8 sorted files to
                               be merged
                                (S or M or both must be specified)

                     O=      - Output dataset (required)

                     DIR=    - Dataset with SORT directves
                               (default: $IN)

                     L=      - Listable output
                               (default: $OUT)
                     L=0     - No listable output

                     ECHO    - Write directives to L=ldn
                               (Not allowed if L=0)

                     RETAIN  - Retain input order for equal keys

                     NOVERF  - Do not verify the sort
                               (default: verify)

        Similar commands:  NOS/BE, NOS:  SORT5
                           VMS:          SORT
```

SPY Generate a histogram on time usage within a program to locate inefficient code.

Syntax:

```
SPY,PREP,BS=bcktsz,D=dbugdn,S=scrtch,
         SUB=rtn1:rtn2:..:rtnn,TS=time.

SPY,POST,ADDRESS,L=listdn,NOLABEL,NOLIB,S=scrtch,
         SUB=rtn1:rtn2:..:rtnn,MINHIT=n.
```

Parameters:

BS= - bucket size in words; each bucket begins on a word address that is a multiple of the bucket size (default: 4)

D= - dataset containing the program's symbol table (default: $DEBUG)

S= - dataset where SPY,PREP will write tables for SPY,POST to use

SUB= - list of up to 20 routines to be analyzed

TS= - time slice in microseconds (default: 500)

ADDRESS - the report will be by address instead of by label

L= - the output report listing dataset (default: $OUT)

NOLABEL - the bucket size will be an entire routine

NOLIB - exclude library calls to routines whose names begin with "$"

MINHIT= - minimum number of hits required to generate a report line for a bucket or label (default: 1; 0 is NOT recommended)

Remarks: At SPY's request, COS reads the address of the current machine instruction. A group of addresses is called a bucket; accessing a bucket is called a hit. After execution, SPY generates a report of all buckets, including a bar graph showing where the time has been spent.

Use SEGLDR to create the absolute; LDR mixes code and data making it more difficult to analyze.

```
Similar commands:  NOS:     HOTSPOT
                   NOS/BE:  SPY; PRINTSPY
                   VMS:     PCA

Examples:    CFT,ON=IZ.  -or-  CFT77,DEBUG.  -or-  CAL,SYM.
                         -or-  PASCAL,O=DM3.
             SEGLDR,CMD='ABS=myabs'.     <-- you must create an
                                             absolute program
             SPY,PREP.                   <-- prepare for SPY
             myabs.                      <-- run your program
             SPY,POST.                   <-- prepare the report

             Since an absolute module is always created, you
             could use

             SEGLDR.
             SPY,PREP.
             $ABD.
             SPY,POST.
```

SUBMIT Send a local dataset to the COS input queue.

```
Syntax:      SUBMIT,DN=dn,SID=sf,DID=df,DEFER,NLRS.

Parameters:  DN=   - Dataset containing the job (required)

             SID=  - Source front-end identifier
                     (2 alphameric characters)

             DID=  - Destination front-end identifier
                     (2 alphameric characters)

             DEFER - Defer the SUBMIT until the dataset is
                     released
                     (default: SUBMIT occurs immediately)

             NLRS  - Do not release the dataset after the
                     SUBMIT; it remains local and read-only
                     (default: dataset is released after the
                             SUBMIT)

Similar commands:  NOS/BE:  BATCH,...,INPUT; ROUTE,DC=IN
                   NOS:     ROUTE,DC=IN; CSUBMIT
                   VMS:     SUBMIT; CRAY SUBMIT

Examples:    SUBMIT,DN=myjobl.
```

```
SWITCH   Turn pseudo sense switches on/off.

         Syntax:      SWITCH,n=x.

         Parameters:  n - switch number (1-6)
                      x - switch position
                          ON  - turned on (set to 1)
                          OFF - turned on (set to 0) .

         Similar commands:  NOS/BE:  SWITCH
                            NOS:     SWITCH; OFFSW; ONSW

         Examples:    SWITCH,2=ON.

UNBLOCK  Convert a blocked dataset to an unblocked dataset.

         Syntax:      UNBLOCK,DN=ldn.            (1)

                      UNBLOCK,I=idn,O=odn.       (2)

         Parameters:  DN= - the dataset to be replaced (using an
                            intermediate dataset $UNBLK)
                            (ldn is rewound before and after)

                      I=  - the blocked input dataset
                            (default: $IN)
                            (idn is not rewound before the copy)

                      O=  - the unblocked output dataset
                            (if previously marked to be unblocked
                            (ASSIGN), odn is not rewound before;
                            otherwise, odn is replaced)

         Remarks:     UNBLOCK is intended primarily for postprocessing
                      datasets created by or for certain stations.

         Examples:    UNBLOCK,DN=myfile.
                                  ^-- Replace MYFILE with unblocked
                                      copy of itself
                      = = = = =
                      UNBLOCK,I=myblk,O=myunblk.
                                  ^-- Copy blocked file MYBLK as
                                      unblocked file MYUNBLK

UPDATE   Source and data maintenance.

         Syntax:       UPDATE,P=pdn,I=idn1:idn2:...:idnn,C=cdn,N=ndn,
                             L=ldn,E=edn,S=sdn,*=m,/=C,DW=dw,DC=dc,
                             ML=n,&,opts.

                       where & is one of: F
                                          Q[=d1:d2:...:dn]
                                          Q='d1,d2,...,dj.dk,...,dn'
```

```
Parameters: P=   - Program library dataset
                   (default: $PL)
            P    - Same as P=$PL
            P=0  - Required for a creation run

            I=   - Input datasets with directives and text
                   (Maximum: 100 datasets)
                   (default: $IN)
            I    - Same as I=$IN
            I=0  - No input dataset

            C=   - Compile output dataset
                   (default: $CPL)
            C    - Same as C=$CPL
            C=0  - No compile output

            N=   - New program library dataset
                   (default: creation run:     $NPL
                             modification run: no new PL)
            N    - Same as C=$CPL
            N=0  - No new PL

            L=   - Listable output
                   (default: $OUT)
            L    - Same as L=$OUT
            L=0  - No listable output

            E=   - Error dataset name
                   (default: $OUT)
            E    - Same as E=$OUT
            E=0  - Errors written to L=ldn
                   (If edn and ldn are the same, ldn is
                    used and E=0)

            S=   - Source output dataset
                   (default: $SR)
            S    - Same as S=$SR
            S=0  - No source output

            *=m  - Master character for directives
                   (defaults: creation run:     *
                              modification run: read from
                                                the PL)

            /=c  - comment character
                   (default: /)

            DW=  - Data width (number of characters written
                   per line to compile and source datasets
                   (defaults: creation run:     72
                              modification run: dw when PL
                                                was created)
            DW   - Same as DW=72 (creation) or use dw when PL
                   was created (modification run)
```

```
                DC= - Declared modifications option:
                        ON  - mod declaration required
                        OFF - mod declaration not required
                       (default: OFF)

                ML= - Message level (highest severity level to
                      suppress):
                        1 - comment
                        2 - note
                        3 - caution
                        4 - warning
                        5 - error
                       (default: 3 - suppress COMMENT, NOTE, and
                                     CAUTION messages)

                F    - Full UPDATE mode
                        (default (F and Q omitted): normal UPDATE
                                                    mode)

                Q=   - Quick UPDATE mode
                        (Maximum: 100 deck names)
                        (default (F and Q omitted): normal UPDATE
                                                    mode)

                opts - NA - no abort
                        NR - no rewind of C and S files
                        IF - write conditional text summary to ldn
                        IN - write input to ldn
                        ID - write identifier summary to ldn
                        ED - write edited card summary to ldn
                        CD - write compile dataset generation
                             directives to ldn
                        UM - write unprocessed modifications to
                             ldn and/or edn
                        SQ - put sequencing in source output in
                             columns dw+1 on (no effect on compile
                             output)
                        NS - no sequencing in compile output
                        K  - sequence decks according to Q

Similar commands:   NOS/BE, NOS:   UPDATE
                    VMS:           CMS; LIBRARIAN

Examples:       UPDATE,I=mysorc,P=0,ID.
                            ^-- create $NPL, list identifiers
                = = = = =
                UPDATE.
                CFT,I=$CPL.
                ...
                /EOF
                *COMPILE a,b,...
                /EOF
```

WRITEDS Initialize a blocked dataset.

Syntax: WRITEDS,DN=dn,NR=nr,RL=rl.

Parameters: DN=dn - required

NR=nr - required - decimal number of records to be written

RL=rl - optional - decimal record length (if non-zero, the first word of each record is the record number as a binary integer starting with 1) (default: 0 (a null record))

Remarks: Writes a single file containing a specific number of records of a specific length. This is useful only for random (direct-access) files, which must be pre-formatted.

Examples: WRITEDS,DN=myfile,NR=1000,RL=125.

***** Appendix C *****

*** DEC VMS DCL Commands ***

DEC VMS DCL (Digital Command Language) commands have the following general syntax:

```
verb        param1  param2  ...            ! comments
@filename   param1  param2  ...  param8    ! comments
RUN         filename                       ! comments
```

verb is the name of the routine to be executed. It consists of an alphabetic character (A-Z, a-z, $, _) followed by 0-31 alphanumeric characters for the name of the command. A procedure (.COM) is executed using an at sign ("@") followed by the name of the procedure file. A user program is executed by the RUN statement.

parami are parameters, which may be positional or keyword.

comments follow an exclamation mark ("!") that is not part of a quoted parameter.

Because VMS has an extensive on-line help facility, the individual DCL commands are not described here. For a list of the help topics, type "HELP". For specific helps, type "HELP topic". The Computer Center maintains the following help libraries which are always available:

HLP$LIBRARY	@CCF	general information about the Computer Center
HLP$LIBRARY_1	@DTLIB	subprograms in library DTLIB (Cray COS, CDC NOS, and DEC VAX/VMS)
HLP$LIBRARY_2	@UTILITIES	commands, programs, procedures, and packages added at DTRC
HLP$LIBRARY_3	@CRAY	DTRC additions to Cray
HLP$LIBRARY_4	@COS	Cray COS JCL statements

*** Selected DEC VAX/VMS Commands ***

The following are a few of the DEC VAX/VMS DCL commands:

ALLOCATE Assign a tape drive to a logical name.

```
Syntax:       ALLOCATE  device  logical_name

Parameters:   device - the logical name of a specific or
                       generic tape drive

              log_name - the name by which the tape is to be
                         known to the job (1-255 characters)

Examples:     $ ALLOCATE  MU:  tape
                           ^-- any tape drive starting with MU
                               will be assigned to logical name
                               TAPE
```

DEALLOCATE Return a previously allocated device and disassociate the job's logical name from the tape drive.

```
Syntax:       DEALLOCATE  logical_name

              DEALLOCATE  /ALL

Parameters:   log_name - the name by which the tape is known
                         to the job

Qualifiers:   /ALL - deallocate all allocated devices

Examples:     $ DEALLOCATE  tape
                           ^-- deallocate the tape drive
                               associated with logical name
                               TAPE
```

DISMOUNT Release a tape volume that was previously mounted.

```
Syntax:       DISMOUNT  device_name

Parameters:   device_name - the physical or logical name of
                            the device to be dismounted

Qualifiers:   /NOUNLOAD - Do not unload the tape (keeps the
                          device and volume in a ready state
                          (default: /UNLOAD)

Examples:     $ DISMOUNT /NOUNLOAD tape
                           ^-- release file TAPE but keep the
                               tape mounted for a future MOUNT
```

INITIALIZE Initialize a magnetic tape.

Syntax: `INITIALIZE device vsn`

Parameters: device - the name given the tape in the ALLOCATE

vsn - a 6-character volume serial number (all DTRC Network tapes are NAnnnn, where nnnn is a 4-digit number)

Remarks: HELP INITIALIZE for additional qualifiers

Examples: See page 6-1-6

MOUNT Mount a magnetic tape and, if labelled, check the label.

Syntax:

```
$ MOUNT  device,...   [ vsn,... ]  [ logical_name ]
         /BLOCKSIZE=mbl  /COMMENT="string"
         /DENSITY=den    /FOREIGN
         /[NO]LABEL      /RECORDSIZE=mrl
         /[NO]UNLOAD     /[NO]WRITE
```

Parameters: device - physical or logical name of the tape drive (for more than one tape, separate with commas or plus signs)

vsn - the volume serial number of the tape(s) (0-6 characters) (not with /FOREIGN)

log_name - the logical name to be used (not needed if is a logical name is used for DEVICE)

Qualifiers: /BLOCKSIZE= - the default block size in bytes (range: 18-65,534; default: 2048)

/COMMENT = - specify additional information to the operator

/DENSITY= - the tape density (1600 or 6250) (default: the density of the first record of the volume)

/FOREIGN - an unlabelled tape

/LABEL - the tape has VAX/VMS ANSI labels
/NOLABEL - the same as /FOREIGN (default: /LABEL)

/RECORDSIZE= - the number of characters in each record - normally used with /FOREIGN and /BLOCKSIZE (mrl <= mbl)

/UNLOAD - unload the tape when DISMOUNTed
/NOUNLOAD - do not unload the tape
(default: /UNLOAD)

/WRITE - the tape can be written
/NOWRITE - the tape is read only
(default: /WRITE)

Examples:

```
$ MOUNT  tape:  /FOREIGN  /DENSITY=1600  -
         /RECORDSIZE=140  /BLOCKSIZE=5040  -
         /comment="Please mount slot98 ", -
                 "vsn=ABCD01 ring"
             ^-- mount a slot tape for writing
                 blocked records
= = = = =
$ mount mytape  NA9999 /density=1600
        /comment="Pls mount with NO ring"
             ^-- mount a read-only tape
= = = = =
```

See page 6-1-6 for an example of initializing a tape.

*** Selected DEC VAX/VMS Additions ***

The following are DTRC additions to DEC VAX/VMS:

```
DETAB   Remove tabs from a file or convert tab-format Fortran source
        lines to fixed-format.

        Syntax:       DETAB  in_file_spec  out_file_spec
                             /TABS=<tab_list>  /INCREMENT=<inc>
                             /FORTRAN

        Parameters:  in_file_spec  - the input file containing tabs

                     out_file_spec - the output file with any tabs
                                     removed
                                     (default: next version of
                                     in_file_spec)

        Qualifiers:  /FORTRAN    - tab-format lines are converted to
                                   fixed-format (the first tab is set
                                   at column 7 (or 6 for continuation
                                   lines) and remaining  tabs are
                                   converted to three blanks)

                                   Since tabs are collapsed to three
                                   blanks, it is unlikely that a
                                   DETABbed line will exceed 72
                                   characters.  If any lines do, you
                                   will be told how many and the
                                   length of the longest line.

                     /NOFORTRAN  - no reformatting is done

                    /INCREMENT=inc   - tabs are set every <inc>
                                       columns

                                       If both /TABS and /INCREMENT
                                       are specified, tabs are set at
                                       the column(s) specified by
                                       /TABS= and every <inc> columns
                                       after that.

                    /LOG             - list summary information and
                                       any warning messages

                                       (Default: /NOLOG)
```

/TABS=n - set one tab at column n

/TABS=(n1,n2,...,nn)
- set tabs at these columns

If /INCREMENT=inc is not specified, then the tabs following the last defined tab stop, are each converted to a single blank.

If /INCREMENT=inc is specified, then the tabs following the last defined tab stop will be every inc columns after the last defined tab stop.

(Defaults: /TABS=0 /INCREMENT=8 /NOFORTRAN)

Note: /FORTRAN overrides /TABS and /INCREMENT.

Remarks: This is useful for:

- Preparing files to go to the Cray, Xerox 8700 or Microfiche, which don't recognize the tab character

- Removing tabs in Fortran programs (for sending to another computer (such as the Cray and CYBER 860) which don't recognize the tab-format).

- Changing the tab values while removing them (e.g., changing from every 8 columns, which is the VAX/VMS standard, to every 5 columns).

Examples: DETAB myprog.for /F

*** Cray Station Commands ***

The VAX/VMS Cray Station provides the VMS user with access to the Cray X-MP.

The following discussion of the Cray station commands is derived from the on-line helps for the CRAY command. Type "CRAY HELP" at the DCL level or "HELP" in Cray context for more detailed information.

```
CRAY     Enter the Cray context utility or executes a single station
         command when that command is supplied as a parameter.

         Syntax:      $ CRAY [station_command]  /BREAKTHROUGH  /REFRESH

         Parameters:  station_command - a single Cray station command
                                          to be executed
                      omitted           - you remain in Cray context until
                                          you enter EXIT

         Qualifiers:  /BREAKTHROUGH - a display refresh occurs during
                                      command input
                                      (valid for refresh mode only)
                                      (default: /NOBREAKTHROUGH)

                      /REFRESH      - enable display refreshing in a
                                      split screen Cray context
                                      (requires DEC_CRT option enabled)
                      /NOREFRESH    - standard teletype environment
                                      (defaults: /REFRESH   (VT100-type
                                                             terminals)
                                                 /NOREFRESH (non-VT100
                                                             terminals))

         See also:    CINT

         Similar commands:  NOS: ICF

         Examples:    $ CRAY
```

CINT From the DCL level, enter a subset of Cray context that accepts only the INTERACTIVE command and its associated subcommands.

Syntax: $ CINT

Remarks: No other Cray context commands are available during a CINT session. CINT is designed to give better interactive performance, since it invokes only a subset of the Cray context image.

For the full set of Cray context commands, use the INTERACTIVE command (in Cray context) instead of CINT.

See also: INTERACTIVE

Similar commands: NOS: ICF

Examples: $ CINT

** Cray Context Commands **

$ Create a temporary VMS subprocess, allowing you to enter DCL commands.

Syntax: $ [dcl_command]

Parameters: dcl_command - any DCL command

Remarks: Since a subprocess is created, any logical names or process resources created in the subprocess will not be available from the main process.

To return to Cray context, type LOGOUT.

Similar commands: NOS ICF:

Examples: $ show users

+ Display the next page of information in Cray context.

Syntax: +

Similar commands: NOS ICF:

Examples: CRAY> +

- Display the previous page of information in Cray context.

Syntax: -

Similar commands: NOS ICF:

Examples: CRAY> -

@ Execute an indirect station command file in Cray context.

Syntax: @file_spec

Parameters: file_spec - a VMS file containing station commands

Remarks: "@" is a synonym for the PLAY command.

See also: PLAY

Similar commands: NOS ICF: /PLAY

Examples: CRAY> @station.COM

ABORT Interrupt the current interactive Cray job step and return control to the COS Control Statement Processor (CSP). CSP will then issue the "!" prompt. Any COS output queued for the terminal will be displayed before the prompt is issued.

Syntax: ABORT

See also: DROP, KILL

Similar commands: NOS ICF: ABORT

Examples: CRAY> ABORT

ATTACH Redirect COS interactive terminal output to an alternate device.

```
Syntax:      ATTACH [alt_device]  /CHAR=(char,pos)
                                  /MRS=max_rec_size
                                  /OFF
                                  /ON
```

```
Parameters:  alt_device - the alternate device
             omitted    - the current output device
```

```
Qualifiers:  /CHAR - route entire record to attached device
                     if character <char> is in position <pos>
                     of the current Cray interactive output
                     record
```

/MRS - route entire record (no carriage control) to attached device if the length of the current Cray interactive output record exceeds max_rec_size

/OFF - do not route Cray interactive records to to attached device (all other parameters or qualifiers ignored)

/ON - enable routing of Cray interactive records to an attached device

Default: /ON

Remarks: The device specified must not be in use and can be any device that accepts record I/O, such as a graphics terminal.

Similar commands: NOS ICF: /CONNECT

ATTENTION Interrupt current interactive Cray job step and enter reprieve processing.

Syntax: ATTENTION

See also: ABORT

Remarks: If reprieve processing not specified, same as ABORT.

Similar commands: NOS ICF: /ATTENTION

Examples: CRAY> ATTENTION

BYE Terminate an interactive session and, optionally, the COS interactive job.

Syntax: BYE /ABORT /SAVE

Qualifiers: /ABORT - terminate the associated COS interactive job

/SAVE - the associated COS interactive job remains active and output is saved; if the job reaches a COS threshold for output messages or requires input, the job is suspended; the terminal can be reconnected to the COS interactive job by the INTERACTIVE command

Remarks: BYE /ABORT is equivalent to QUIT.

See also: QUIT

Similar commands: NOS ICF: /BYE, /LOGOFF, /QUIT

Examples: CRAY> BYE

CLEAR Terminate any display command and clears the display portion of the screen.

Syntax: CLEAR

Remarks: CLEAR is only available when Cray context is in refresh mode.

Examples: CRAY> CLEAR

COLLECT Store COS interactive output in a VMS file.

Syntax: COLLECT file_spec /ECHO /OFF /ON

Parameters: file_spec - the VMS file to receive the COS interactive output

Qualifiers: /ECHO - display the output generated at the terminal as well as the VMS file
/NOECHO - do not echo the generated output at the termiinal; only into the VMS file (default: /ECHO)

/OFF - stop writing COS job output to a VMS file and close the VMS file (ignore other qualifiers

/ON - write COS job output to a VMS file (default: /ON)

Remarks: COLLECT can be used before the interactive job is initiated.

Examples: CRAY> COLLECT mycosfile.out

COMMENT Insert comments into an indirect station commrnd file stream.

Syntax: COMMENT string

Parameters: string - any text

Remarks: The comment line can be 256 characters long, including "COMMENT".

See also: @, MESSAGE

Similar commands: NOS ICF: /*

Examples:

```
COMMENT This is a comment
```

CONTROL_Z CTRL-Z (^Z) exits the current processing mode.

Syntax:

```
^Z          <-- ^ is the CTRL key
```

Remarks: In response to the Cray context prompt (CRAY>), you are returned to DCL; in a Cray interactive session, you are returned to command mode. While you are being prompted for command parameters, CTRL-Z cancels the command.

CTRL-Z also terminates the execution of an indirect station command file.

See also: @

Examples:

```
! ^Z           <-- leave Cray session
CRAY> QUIT     <-- terminate Cray session
CRAY> ^Z       <-- terminate Cray context
$              <-- you are back at the DCL level
```

DATASET Test for the existence of a COS permanent dataset.

Syntax:

```
DATASET  pdn  /ID=id  /ED=ed  /OV=owner
```

Parameters: pdn - name of PDS

Qualifiers:

/ID= - id of the dataset (1-8 characters)
(default: null)

/ED= - edition number of the dataset (1-4095)
(default: current highest edition number)

/OV= - owner of the dataset

Examples:

```
DATASET,myfile.
```

DELAY Suspend execution of an indirect station command file for a specified period of time.

Syntax:

```
DELAY  seconds
```

Parameters: seconds - suspension time in seconds

Examples:

```
DELAY 20
```

DISCARD Discard all output from a COS interactive session until the next COS prompt is issued.

Syntax: DISCARD

Similar commands: NOS ICF: /DISCARD

Examples: DISCARD

DROP Terminate a COS job and return the associated output dataset. COS job execution enters reprieve processing after the next COS EXIT control statement.

Syntax: DROP jsq

Parameters: jsq - job sequence number

Remarks: Use STATUS to obtain the job sequence number (COS jsq).

KILL terminates the job immediately; DROP continues processing after an EXIT statement.

See also: ABORT, KILL

Examples:

```
$ CRAY
CRAY> STATUS
CRAY> DROP 9876
```

EOF Sends an end-of-file record to a connected COS interactive job.

Syntax: EOF

Remarks: EOF is normally required to terminate COS file input from the terminal.

Similar commands: NOS ICF: /EOF

Examples: CRAY> EOF

EXIT Leave Cray context command mode and return to DCL.

Syntax:

```
EXIT
^Z
```

Remarks: EXIT will close the file specified in a RECORD command, if it is still open.

See also: RECORD

```
         Similar commands:  NOS ICF:  /EXIT

         Examples:     CRAY> EXIT


HELP     Display help information on the Cray station commands.

         Syntax:       HELP  [station_command]

         Parameters:   station_command - a specific command for which
                                         help is desired
                       omitted         - a list of all available commands

         Similar commands:  NOS ICF:  /HELP

         Examples:     $ CRAY HELP
                       - - - - -
                       CRAY> HELP
                       - - - - -
                       CRAY> HELP INTERACTIVE


INTERACTIVE  Initiate or restart an interactive session.

         Syntax:       INTERACTIVE  /JN=jobname
                                    /LOWER
                                    /MML=maximum_message_length
                                    /UPPER
                                    /US=username

         Qualifiers:   /JN=    - the COS interactive jobname (1-7 chars)
                                 (if omitted, you will be prompted for it)

                       /LOWER  - don't convert lower case to upper case
                                 (default: /LOWER)

                       /MML=   - the maximum message length

                       /UPPER  - convert lower case to upper case
                                 (default: /NOUPPER (/LOWER))

                       /US=    - the COS username (1-15 characters)
                                 (if omitted, you will be prompted for it)

         Remarks:      Cray interactive is available only an attached
                       station.

         See also:     CINT

         Similar commands:  NOS ICF:  /LOGON

         Examples:     $ CRAY INTER
                       - - - - -
                       $ CRAY
                       CRAY> INTER /JN=jobname /US=xxxx
```

ISTATUS Get the status of your COS interactive job (with CPU time used and the last COS logfile message).

Syntax: ISTATUS

See also: JSTAT, STATUS

Examples: ISTATUS

JOB Display the status of a specific COS job.

Syntax: JOB jobname /JSQ=jsq

Parameters: jobname - the COS job name

Qualifiers: /JSQ= - the job sequence number from which to start the search for the job

Similar commands: NOS ICF: /STATUS

Examples: JOB myjob4

JSTAT Display the status of a specific job and its related tasks.

Syntax: JSTAT jsq /[NO]CYCLE /[NO]TRANSLATE

Parameters: jsq - the job sequence number

Qualifiers: /CYCLE - cycle the display refresh through all the available information

/NOCYCLE - display only the current page until you enter "+" or "-"
(default: /NOCYCLE)

/TRANSLATE - display the terminal ID field in the VMS UIC equivalent

/NOTRANSLATE - display it in the station internal form
(default: /TRANSLATE)

Remarks: Use STATUS to obtain the COS job sequence number (jsq).

See also: ISTATUS, STATUS

Similar commands: NOS ICF: /STATUS

Examples: JSTAT

KILL Delete a job from the input queue, or immediately terminate an executing job, or delete the job's output dataset from the output queue.

Syntax: KILL jsq

Parameters: jsq - the job sequence number

Remarks: Use STATUS to obtain the COS job sequence number (jsq).

KILL terminates the job immediately; DROP continues processing after an EXIT statement.

See also: ABORT, DROP

Similar commands: NOS ICF: /ABORT

Examples:

```
CRAY> STATUS
CRAY> KILL 9876
```

LOGFILE Provides access to the station logfile messages.

Syntax:

```
LOGFILE  [file_spec]  /ACQUIRE  /ALL
                      /BEFORE=time  /DISPOSE
                      /ERROR  /INTERACTIVE
                      /JOB  /MASTER  /NETWORK
                      /NODE=nodename  /[NO]NOTIFY
                      /OPERATOR  /OUTPUT=file_spec
                      /PRINT  /RELEASE
                      /SINCE=time  /SUCCESS
                      /STMSG  /TRANSLATE
```

Parameters: file_spec - An alternate station logfile to be displayed

Qualifiers: /ACQU - display ACQUIRE and FETCH messages

/ALL - display all messages

/BEFO - display messages from before a specified time

/DISP - display DISPOSE messages

/ERRO - display error messages

/INTE - display interactive processing messages

/JOB - display job submission messages

/MAST - display COS master operator messages

/NETW - display DECnet messages (all nodes)

/NODE= - display DECnet messages (one node)

/NOTI - you will be notified an asychronous LOGFILE operation is performed (requires /RELEASE) (default: /NONOTIFY)

/OPER - display operator messages

/OUTP= - VMS file to receive station messages currently being displayed

/PRIN - print station messages currently being displayed

/RELE - close the existing logfile and create a new version

/SINC= - display messages since a specified time

/SUCC - display success, warning, and informational messages

/STMS - display COS station messages and associated replies

/TRAN - display terminal ID field (TID) as the VMS UIC equivalent
/NOTR - display TID in the station internal form (default: /TRANSLATE)

Examples: CRAY> LOGFILE jobname.LOG /SINCE=09:15

LOOP Restart execution of an indirect station command file at the beginning.

Syntax: LOOP

Remarks: CTRL-Z must be issued to terminate looping.

Examples: CRAY> LOOP

MESSAGE Send a message to the COS job logfile.

```
Syntax:       MESSAGE  string  /JN=jobname
                               /JSQ=jsq
```

Parameters: string - the message text (for embedded blanks, enclose in quotes ("..."))

Qualifiers: /JN= - the name of the COS job to receive the message (requires /JSQ)

/JSQ= - the job sequence number of the COS job to receive the message

See also: COMMENT

Similar commands: NOS ICF: /*

Examples: MESSAGE This is a message

PAUSE Suspend execution of an indirect station command file.

Syntax: PAUSE

Remarks: Control passes to the terminal, where you can terminate the command file by entering a command or resume it by entering a null line (<RET>).

Examples: PAUSE

PLAY Execute an indirect station command file in Cray context.

Syntax: PLAY file_spec

Parameters: file_spec - a VMS file containing station commands

Remarks: PLAY files cannot themselves contain other (embedded) PLAY commands.

"@" is a synonym for the PLAY command.

Similar commands: NOS ICF: /PLAY

Examples: CRAY> PLAY station.COM

QUIT Terminate a Cray interactive session and the corresponding COS interactive job.

Syntax: QUIT

Remarks: QUIT is the equivalent of BYE /ABORT.

See also: BYE

Similar commands: NOS ICF: /BYE, /LOGOFF, /QUIT

Examples:

```
!^Z             <-- leave Cray session
CRAY> QUIT      <-- terminate the Cray session
CRAY> EXIT      <-- terminate the Cray station
```

RECORD Start or stop the recording of terminal input to a file while in Cray context for later use with the PLAY or @ commands.

Syntax: RECORD [file_spec] /ON /OFF

Parameters: file_spec - the file into which terminal input is to be recorded

Qualifiers: /ON - start command recording (file_spec required)

/OFF - end command recording (default: /ON)

Remarks: Exiting Cray context automatically issues a RECORD/OFF.

Examples:

```
RECORD station.com  /ON
   ...
RECORD /OFF
```

RELEASE Releases a dataset that is held by COS.

Syntax: RELEASE jsq

Parameters: jsq - the job sequence number

Remarks: The dataset status must be HOLDING. This may be due to VAX disk quota limitations.

Use STATUS to obtain the COS job sequence number (jsq).

Examples:

```
CRAY> STATUS
CRAY> RELEASE 9876
```

REMOVE Delete entries in the dataset staging queue.

Syntax: REMOVE queue_id /LOCKED /SPOOL /STAGE

Parameters: queue_id - an 8-character hexadecimal number from the SHOW QUEUES display (leading zeros can be omitted)

Qualifiers: /LOCKED - controls whether or not locked entries are removed (default: /NOLOCKED)

/SPOOL - remove an entry in the network spooled dispose queue

/STAGE - remove an entry in the Cray staging queue

RERUN Immediately end the processing of a COS job and put it back into the input queue.

Syntax: RERUN jsq

Parameters: jsq - the job sequence number

Remarks: The job input dataset is saved and all output datasets associated with the job are deleted. The job input dataset is then rescheduled so the job can be rerun. No action is taken if the job execution is complete or if COS determines the job cannot be rerun.

Use STATUS to obtain the COS job sequence number (jsq).

SAVE Stages a VMS file to COS disk storage.

Syntax:

```
SAVE  file_spec  /DELETE  /DF=d  /ED=ed  /ID=id
                 /MN=mn  /PDN=pdn  /RD=rd
                 /RT=rt  /US=us  /WT
```

Parameters: file_spec - the file to be staged

File_spec qualifiers:

```
/DELE  - delete the file when is has been
         successfully staged to the Cray

/DF=   - dataset format: CB, BB, or TR
         (default: CB)

/ED=   - edition number (0-4095)
         (default: next higher number)

/ID=   - identification (1-8 alphameric chars)

/MN=   - maintenance control word

/PDN=  - dataset name to be used
         (converted to uppercase)
         (default: the input file name)

/RD=   - read permission control word

/RT=   - the retention period, in days
```

```
                /US=   - the COS username

                /WT=   - the write permission control word

     Examples:  SAVE myfile.dat /PDN=mydata /US=ABCD
```

SET TERMINAL Define the terminal working environment.

SET TERMINAL FORTRAN
SET TERMINAL NOFORTRAN

Specify whether the terminal is to interpret output records from a COS interactive session as having FORTRAN carriage control.

Default: NOFORTRAN

SET TERMINAL INFORM
SET TERMINAL NOINFORM

Enable/disable the sending of station messages to the user logged on to VMS at a VAX terminal.

Default: NOINFORM

SET TERMINAL PAGE
SET TERMINAL PAGE=lines
SET TERMINAL NOPAGE

Specify the number of lines of output before a page break.

Default: NOPAGE

Default for lines: determined by the scroll setting

```
SET TERMINAL REFRESH
SET TERMINAL REFRESH=seconds          <-- integer 0-60
SET TERMINAL NOREFRESH
```

REFRESH provides a split-screen Cray context environment and is supported only on terminals with the DEC_CRT attribute. NOREFRESH provides a line-by-line Cray context environment.

```
Defaults: REFRESH   (VT100-type terminals)
          NOREFRESH (non-VT100-type terminals)
```

SET TERMINAL SCROLL=lines

Changes the Cray context window size.

"lines" is the size of the command area (bottom window) and must be an integer from 3 to 13.

Default for lines: 4

SET TERMINAL WIDTH=80
SET TERMINAL WIDTH=132

Changes the width of the terminal within Cray context.

Default: 80

SHOW QUEUES Display entries in the dataset staging queue.

```
Syntax:      SHOW  QUEUES  /ACQUIRE  /ALL  /CYCLE  /JOB
                           /NODE=node_id  /OWNER  /SAVE
                           /STAGE  /TRANSLATE

Qualifiers:  /ACQU  - display all entries originating from
                      COS (ACQUIRE or FETCH)
                      (default: /ALL)

             /ALL   - display all entries
                      (same as /ACQUIRE/JOB/SAVE)
                      (default: /ALL)

             /CYCL  - cycle the display refresh through all
                      the available information
             /NOCYC - display only the current page until you
                      enter "+" or "-"
                      (default: /NOCYCLE)

             /JOB   - display entries originating from VMS
                      (default: /ALL)

             /NODE= - display entries from a specific DECnet
                      node
                      (valid only from an attached station)

             /OWNER - display only your entries

             /SAVE  - display entries for SAVEd datasets
                      (default: /ALL)

             /STAGE - display all Cray staging entries

             /TRAN  - display the terminal ID field in the VMS
                      UIC equivalent
             /NOTRA - display it in the station internal form
                      (default: /TRANSLATE)
```

Remarks: The following fields are displayed:

- Position in the staging queue (L is a locked entry i.e., one that is being processed)
- Request type (JB=job, AC=acquire/fetch, SV=save)
- Queue ID for use in the REQUEUE and RELEASE commands

```
                       . VAX username of entry owner
                       . Dataset transfer name (job name or dataset
                         name)
                       . Dataset terminal ID (TID)

          Similar commands:  NOS ICF:  /STATUS

          Examples:    SHOW QUEUES /OWNER
                                    ^-- display all your entries


SNAP      Copy the current contents of the display region into a VMS
          file.

          Syntax:      SNAP  file_spec  /[NO]ESCAPE

          Parameters:  file_spec - VMS file to receive the snapshot

          Qualifiers:  /ESCAPE   - retain escape sequences
                       /NOESCAPE - remove escape sequences
                                   (default: /NOESCAPE)

          Remarks:     In line-by-line mode, the last display requested
                       is recorded.


          Examples:    SNAP snap.job123


STATCLASS  Display the current COS job class structure.

          Syntax:      STATCLASS  /[NO]CYCLE

          Qualifiers:  /CYCLE       - cycle the display refresh through
                                      all the available information
                       /NOCYCLE     - display only the current page until
                                      you enter "+" or "-"
                                      (default: /NOCYCLE)

          Similar commands:  NOS ICF:  /ICFSTATUS, /STATUS

          Examples:    STATCLASS


STATUS    Displays the COS system status.

          Syntax:      STATUS  /ALL  /CLASS=class_id  /CYCLE  /EXECUTING
                               /HOLD  /ID=mainframe_id  /INPUT
                               /NODE=node_id  /OUTPUT  /OWNER
                               /RECEIVING /SENDING   /TRANSLATE /VAX

          Qualifiers:  /ALL   - display all COS jobs
                       /CLAS= - display jobs and datasets of a specific
                                job class
                                (default: /ALL)
```

/CYCL - cycle the display refresh through all available information
/NOCY - display only the current page until you enter "+" or "-"
(default: /NOCYCLE)

/EXEC - display the execution queue status
(default: /EXECUTION)

/HOLD - display COS datasets in the hold queue

/ID= - display jobs and datasets originating from a specific mainframe

/INPU - display the input queue status

/NODE= - display the entries for a specific DECnet node

/OUTP - display the output queue status

/OWNE - display only your jobs and datasets

/RECE - display the Cray receiving queue status
(default: /RECEIVING)

/SEND - display the Cray sending queue status
(default: /SENDING)

/TRAN - display terminal ID field (TID) as the VMS UIC equivalent
/NOTR - display TID in the station internal form
(default: /TRANSLATE)

/VAX - display only COS jobs related to this VAX/VMS station (or network of stations)

See also: ISTATUS, JSTAT

Similar commands: NOS ICF: /STATUS

Examples: STATUS

SUBMIT Stage a VMS file to the COS input queue.

Syntax: SUBMIT file_spec /AFTER=time /EOF=eof /PRINT

SUBMIT f1,f2,... /AFTER=time /EOF=eof /PRINT

Parameters: file_spec - single VMS file with a complete COS job

f1,f2,... - two or more files to be combined to create a complete COS job

Qualifiers: /AFTER= - specify when the job is to be sent to the Cray

/EOF= - specify what represents an end-of-file (e.g., /EOF="E O F") (default: /EOF="/EOF")

/PRINT - print the job's output file on COS job completion

/NOPRINT - put the COS job's output into your VMS file COS_jobname.CPR

(default: /NOPRINT)

Remarks: The file must contain a COS job. By default, the job's output (including the dayfile) is sent to the originating directory.

Similar commands: NOS: CSUBMIT

Examples:

```
CRAY> SUBMIT myjob1
   -or-
$ CRAY SUBMIT myjob1
= = = = =
CRAY> SUBMIT myjob2,myprog2.for,mydata2.dat
   -or-
$ CRAY SUBMIT myjob2,myprog2.for,mydata2.dat
```

SWITCH Set or clear COS job sense switches.

Syntax:

```
SWITCH  jsq  ssw  /OFF
SWITCH  jsq  ssw  /ON
```

Parameters: jsq - the COS job sequence number

ssw - the sense switch number (1-6)

Qualifiers: /OFF - turn switch <ssw> off

/ON - turn switch <ssw> on

Remarks: These switches can be used for program synchronization on the Cray.

Examples:

```
CRAY> STATUS                 <-- to get the jsq
CRAY> SWITCH 9876 3 /ON      <-- turn on switch 3
```

***** Appendix D *****

*** CDC NOS JCL Commands ***

CDC NOS JCL commands have the following general syntax:

```
  verb,param1,param2,....    comments
  verb(param1,param2,...)    comments
```

verb is the name of the routine to be executed. It consists of an alphabetic character (A-Z) followed by 0-6 alphanumeric characters for the name of the command.

parami are parameters, which may be positional or keyword.

comments follow the terminator (a period "." or right parenthesis ")").

*** Strings ***

The following string representations are used in this appendix:

aa...a	1 or more alphabetic characters
axx...x	1 or more alphanumeric characters, the first alphabetic
xxx...x	1 or more alphanumeric characters
nn...n	1 or more decimal (unless otherwise stated) digits
nn...nB	1 or more octal digits
nn...nD	1 or more decimal digits

*** Some Common Parameters ***

The following parameters are used in many JCL commands. If they have a different meaning or a special condition, it will be mentioned in the individual description.

CM=nnnnnn Maximum central memory field length, octal unless D suffix or 8 or 9 in number, maximum 376500

CT=ct File permit Category Type

ct	meaning
P or PR or PRIVATE	private
S or SPRIV	semiprivate
PU or PUBLIC	public

JSN=jsn Job Sequence Name (aaaa)

lfn Local File Name (xxxxxxx, 7 maximum)
(lfn's starting with ZZ are reserved to NOS)

lfn_in Input local file name
(normal default is INPUT)

lfn_out Output local file name
(normal default is OUTPUT)

L=lfn Output listing file (xxxxxxx, 7 maximum)
(normal default is OUTPUT)

M=m

m File permissions (file access mode)

m	meaning
E (EXECUTE)	you can execute; others can read or execute concurrently
R (READ)	all can read or execute concurrently
RU (READUP)	all can read or execute; one (other) user can rewrite the file
RA (READAP)	all can read or execute; one (other) user can lengthen the file
RM (READMD)	all can read or execute; one (other) user can lengthen or rewrite the file
U (UPDATE)	all can read or execute; you can rewrite the file
A (APPEND)	all can read or execute; you can lengthen the file
M (MODIFY)	all can read or execute; you can lengthen or rewrite the file
W (WRITE)	you can read, execute, lengthen, rewrite, or shorten the file; others have no concurrent access

NA	No abort on errors
non-null	Indicates that the parameter the presence of a string of characters (normally xxxxxxx, 7 maximum)
pfn	Permanent File Name (xxxxxxx, 7 maximum) (normal default is the lfn)
PW=password	password for access to another user's permanent file
UJN=ujn	User Job Name (xxxxxxx, 7 maximum)
UN=un	User Name (xxxxxxx, 7 maximum)

*** Summary of CDC NOS JCL Commands ***

The following are NOS JCL statements, except as indicated by:

(DTRC) A command, procedure or program added at DTRC

(IAF) InterActive Facility

(ICF) Interactive Cray Facility (ICF);
begin with a slash (/), not to be confused with the IAF prompt

JCL statements for certain NOS features are indicated by:

(CRM) Cyber Record Manager

(Loader) Loader control statements

* Entire line is a comment.

Syntax: *comment

See also: COMMENT

Similar commands: Cray: *
NOS/BE: COMMENT
VMS: !

Examples: * This is a comment ---

ct1 (IAF) Interrupt the current job step (user-break-1).

Syntax: ct1

Parameters: ct - the network control character
(normally percent (%))

Remarks: Some terminals require that ^S be entered before and ^Q after this command.

Examples: %1

ct2 (IAF) Terminate the current job step (user-break-2); cancel the output in progress.

Syntax: ct2

Parameters: ct - the network control character
(normally percent (%))

Remarks: Some terminals require that ^S be entered before and ^Q after this command.

Examples: %2

ctD (IAF) Immediately detach a terminal job from the terminal.

Syntax: ctD

Parameters: ct - the network control character (normally percent (%))

Remarks: To detach during output, interrupt the output, then enter ctD.

Any type-ahead commands are discarded.

Examples: %D

ctE (IAF) Immediate detailed job status.

Syntax: ctE

Parameters: ct - the network control character (normally percent (%))

Examples: %E

ctS (IAF) Immediate abbreviated job status.

Syntax: ctS

Parameters: ct - the network control character (normally percent (%))

Remarks: Response is one of: EXECUTE, IDLE (waiting for you), or WAIT (waiting for system resources).

Examples: %S

ACCESS (IAF) Select the ACCESS subsystem.

Syntax: ACCESS

Remarks: Required to communicate with another interactive terminal (DIAL, WHATJSN)

RUN will not work in the access subsystem.

Examples:

```
ACCESS.
WHATJSN.
DIAL,jsn,message.
```

APPEND Append information to the end of an indirect file without retrieving the file.

Syntax: `APPEND,pfn,lfn_1,lfn_2,...,lfn_n/UN=un,PW=pw,NA,WB.`

Parameters: UN=un - required only for files in another catalog

PW=pw - Required for files with passwords in another catalog

WB - wait for a busy file

Remarks: You cannot append to a direct file.

You cannot append to a direct access (random) file.

Similar commands: NOS/BE: EXTEND (for attached PF)
VMS: APPEND

Examples: `APPEND,myperm,new1,nu2.`

APPSW *(IAF) Switch temporarily to an altername NAM application* program.

Syntax: `APPSW,AP=appl,Z.data`

`APPSW,appl,Z,data`

Parameters: appl - a NAM application

appl	meaning
IAF	InterActive Facility
ICF	Interactive Cray Facility
RBF	Remote Batch Facility

Z - any characters following the terminator are passed to the secondary application as data

data - first 50 characters after the terminator are passed to the secondary application

Examples: `APPSW,ICF.` <-- go into Cray Interactive Facility

ASCII (IAF) Set terminal to ASCII.

Syntax: ASCII

Remarks: ASCII support is 128 characters (95 graphics plus 33 controls) in 6/12-bit display code.

See also: CSET, NORMAL

Examples: ASCII

ASSIGN Assign a file to a device.

Syntax: ASSIGN,nn,lfn,ckpt.

Parameters: nn - device or device type for assignment

type	equipment
MS	mass storage device (a disk)
MT	7-track magnetic tape
NE	null equipment
NT	9-track magnetic tape
TT	interactive terminals (only interactive origin jobs)

lfn - the file to be assigned

ckpt - lfn is to be a checkpoint file

ckpt	meaning
CK	put each dump at end of lfn
CB	put each dump at beginning of lfn

See also: FILE

Similar commands: Cray: ACCESS
NOS/BE: CONNECT; DISCONT; REQUEST

Examples:

```
ASSIGN,MS,OUTPUT.      <-- direct output to a disk
                           file until returned or
                           reassigned

ASSIGN,TT,XYZ.         <-- assign file XYZ to
                           your terminal
```

ATTACH Assign a direct permanent file to a job.

Syntax:

```
ATTACH,lfn_1=pfn_1,lfn_2=pfn_2,...,lfn_n=pfn_n
      /M=m,UN=un,PW=pw,NA,RT,WB.
```

Parameters: lfn_i=pfn_i - if lfn_i is omitted, pfn_i is used;
if lfn_i exists, it is discarded

M=m - (default: M=READ)

UN=un - required for a file in another catalog
PW=pw - required if UN is specified and the file has a password
RT - real-time processing (job continues after requesting staging from the MSS - a second ATTACH is required to access the staged file; if no staging is necessary, the file is assigned immediately)
WB - wait for a busy file

See also: GET, FETCH (Cray)

Similar commands: Cray: ACCESS; ACQUIRE
NOS/BE: ATTACH
VMS: no local file concept

Examples:

```
ATTACH,MYFILE/M=W.
                ^-- allows file MYFILE to be
                    overwritten (such as after
                    editing)
```

BASIC (IAF) Select the BASIC subsystem.

Remarks: Use FSE and X,BASIC.

BASIC See X,BASIC to compile a BASIC program without entering the BASIC subsystem.

BATCH (IAF) Select the BATCH subsystem.

Syntax: BATCH,fl

Parameters: fl - initial running field length for subsequent commands
(default: 0)

Remarks: This is the default subsystem when you enter IAF.

Examples: BATCH

```
BEGIN     Transfer control to a procedure.

          Syntax:       BEGIN,pname,pfile,p1,p2,...,pn.comment          (1)

                        -pname,pfile,p1,p2,...,pn.comment               (2)

                        pname,p1,p2,...,pn.comment                      (3)

                        pfile,p1,p2,...,pn.comment                      (4)

          Parameters:   pname   - the name of the procedure to be executed
                                  (default: the procedure at the current
                                            position in the file)

                        pfile   - the file containing the procedure
                                  (default search order:
                                     1) local file PROCFIL
                                     2) your permanent file PROCFIL
                                     3) public-access procedure file
                                        PROCFIL/UN=LIBRARY)

                                  (note: the search order is different
                                         from NOS/BE, which does not
                                         include item 2)

                                  (note: if you have a local or permanent
                                         file PROCFIL, you cannot access
                                         any of the public procedures,
                                         therefore, use another filename
                                         for your procedures)

                        pi      - an optional parameter

                        comment - value associated with the CK keyword in
                                  the procedure header

          Remarks:      Except for interactive execution of a procedure,
                        the BEGIN statement may be continued on more than
                        one line.

                        For interactive procedures, a question mark "?"
                        may be used:
                          . for a list of parameters:
                              BEGIN,pname,pfile,?
                          . for help with a parameter:
                              BEGIN,pname,pfile,p1,...,pi?
                          . for help during interactive processing:
                              param?
```

To accept the default value for a parameter during interactive processing, press the carriage return.

To accept the default value for the current and all remaining parameters, enter a terminator. If any required parameters remain undefined, you will be prompted for them.

See also: REVERT

```
Similar commands:  Cray:    CALL
                   NOS/BE:  BEGIN
                   VMS:     @filename
```

```
Examples:    GET,MYPF,MYPROCFILE/UN=xxxx.
             BEGIN,MYPROC,MYPF.
                           ^-- a user procedure

             BEGIN,NEWCHRG.
                           ^-- a public-access procedure in
                               PROCFIL/UN=LIBRARY
```

BELOAD Selectively load files from a NOS/BE tape created by DUMPF or BEGIN,SELDUMP.

```
Syntax:      BELOAD,I=i,L=l,T=t,OP=op.

Parameters:  I=  - input file of directives
                   (default: INPUT)

             T=  - local file name of the tape
                   (default: TAPE)

             OP= - load option for directives with PF=
                      op   meaning
                      --   -------
                      N    normal restore (don't replace
                           existing file)
                      R    replace specified file if it already
                           exists
                   (default: N)

Directives:  ID=id,FN=fn,PF=pf,CY=cy,CT=ct,TY=ty,PW=pw,M=m.

             ID= - the NOS/BE ID (required)
             FN= - the NOS filename
                   (also requires PF=)
             PF= - the NOS/BE filename (consisting of letters
                   and digits only!)
```

CY= - the NOS/BE cycle number
(default: the highest cycle only)

TY= - the NOS file type

ty	meaning
D	direct
I	indirect

(default: D)

PW= - the NOS password for the file

M= - the NOS access mode
(default: R)

Remarks: BELOAD is in library BETONOS/UN=LIBRARY.

Similar commands: NOS/BE: LOADPF, BEGIN,SELLOAD (DTRC)
VMS: BACKUP

Examples:

```
ATTACH,BETONOS/UN=LIBRARY.
LIBRARY,BETONOS/A.
LABEL,TAPE,VSN=NA9876,D=GE,F=S,LB=KL,R,PO=R.
BELOAD,I=fylist.
UNLOAD,TAPE.
LIBRARY,BETONOS/D.
```

where fylist contains:

```
  ID=abcd,FN=myfyl1,PF=mynosbefyl1.
  ID=abcd,FN=myfyl2,PF=mynosbefyl2,CY=423,TY=I.
  ID=abcd,FN=myfyl3,PF=mynosbefyl3,CT=PU.
```

BKSP Backspace a file (by logical records).

Syntax: BKSP,lfn,n,m.

Parameters: lfn - the file to be backspaced

n - decimal number of logical records to backspace
(Default: 1; max: 262143)

m - file mode: C (coded) or B (binary)
(Default: B)

Similar commands: Cray: SKIPF, SKIPR
NOS/BE: BKSP, SKIPB

Examples: BKSP,myfile,4.

BLANK Blank label a magnetic tape.

Syntax:

```
BLANK,VSN=vsn,MT¦NT,D=den¦den,CV=cv,FA=fa,
      OFA=ofa,VA=va,OWNER=username/familyname,
      LSL=lsl,U.
```

Parameters:

VSN= - 1- to 6-character volume serial number (don't use a current local file name or the file will be lost)

MT - 7-track tape

NT - 9-track tape

D=den
den - tape density

MT		NT	
den	density	den	density
LO	200 cpi	HD	800 cpi
HI	556 cpi	PE	1600 cpi
HY	800 cpi	GE	6250 cpi
200	200 cpi	800	800 cpi
556	556 cpi	1600	1600 cpi
800	800 cpi	6250	6250 cpi

CV= - conversion mode for 9-track labels (do not use with MT)

cv	meaning
AS	ASCII/6-bit display code
US	same as AS
EB	EBCDIC/6-bit display code

FA= - File accessibility character

fa	meaning
blank	unlimited access
A	only the owner can access it
other	future accesses must specify this character

OFA= - old file accessibility character when relabelling a tape with one

VA= - volume accessibility character

va	meaning
blank	unrestricted access
other	only a system job can destroy the VOL1 label

(must always be a labelled tape)

OWNER= - ownership identification - determines file accessibility (FA)

LSL= - label standard level entered in VOL1 label

lsl	meaning
1	tape labels and data format for this volume are ANSI std
blank	may or may not be ANSI

(Default: 1)

U - unload after blank labelling
(Default: do not unload)

Similar commands: Cray: no tapes
NOS/BE: console operator command
VMS: MOUNT

Examples: `BLANK,VSN=NA9999,D=GE.`

BLOCK Add one or more lines of 10x10 block letters to a file.

Syntax: `BLOCK,lfn,rewind,cc./line_1/line_2/.../line_n`

Parameters: rewind - rewind option
R - rewind lfn before writing
NR - do not rewind lfn
(Default: NR)

cc - carriage control character to be inserted before the first line
(Default: 1)

/ - a delimiter character which separates the lines in the command - may be any character and must immediately follow the terminator - successive delimiters generate blank lines

line_i - a string of up to 10 characters for one line of blocked characters - or one of the following:
DATE - current date
TIME - current time
USER - current user name
UJN - user job name
JSN - job sequence number

Similar commands: NOS/BE: BANNER, BANNER3, BANNER6, BANNERS (all DTRC)
VMS: VSYS:BANNER, VSYS:BANNER6

Examples: BLOCK,blockf.*myjob*date <-- * is the delimiter

BYE (IAF) Terminate an application.

Syntax: BYE,appl

Parameters: appl - if IAF is your primary application, appl is one of: IAF, RBF (see APPSW)

if IAF is your secondary application, appl is one of:

appl	meaning
omitted	end IAF and return to primary application
ABORT	end both primary and secondary applications

(anything else will be treated the same as if omitted)

See also: GOODBYE, LOGOUT; HELLO, LOGIN

Similar commands: Cray: ^Z, QUIT
NOS/BE, VMS: LOGOUT

Examples: BYE

CATALOG List information about each record in a file.

Syntax: CATALOG,lfn,p_1,p_2,...,p_n.

Parameters: lfn - the file to be cataloged.

```
pi  - parameters
        N=n     - catalog n files
                      N=0     - catalog until
                                double EOF
                      N       - catalog until EOI
                      omitted - same as N=1

        L=name - the output file
                      omitted - same as L=OUTPUT
```

T - list entire text for records starting with APRD, CMRD, EQPD, IPRD, LIBD

U - list all records in a user library
omitted - list ULIB record

D - suppress comments field and all but first page heading

CS - suppress character set indicator (63/64) for OPL and OPLC records

R - rewind lfn before and after

Remarks: Don't use on S, L or F tapes. COPY them to a disk file, or to an I or SI tape before CATALOGing.

For terminal output, set to NORMAL mode.

See also: ITEMIZE

Similar commands: Cray, NOS/BE: ITEMIZE

Examples: CATALOG,myfile,N,R.

CATLIST List permanent file information.

Syntax: CATLIST,LO=lo,FN=pfn,UN=un,NA,L=lfn,PW.

Parameters: LO=lo - list options

lo	meaning
F	all information about one or all files (3 lines per file)
FP	access permissions of a file
0	(zero) alphabetical list of names of indirect and direct files
P	list only names of users who have access to a private file or who have accessed a semiprivate file
X	LO=F plus security access categories for one file

(default: 0)

FN=pfn - a single file specification (required for LO=FP, LO=P, and LO=X) - one or more single-character wildcards (*) are allowed (e.g., ABC****)

```
                    L=lfn   - file to receive the CATLIST output
                                (Default: L=OUTPUT)

                    PW      - display passwords in LO=F output

        Remarks:    In the CATLIST output, a filename enclosed in
                    parentheses means the file is on the MSS.

        Similar commands: Cray:    AUDIT
                          NOS/BE:  AUDIT;    BEGIN,AUDIT;
                                   MSAUDIT;  BEGIN,MSAUDIT
                          VMS:     DIRECTORY

        Examples:   CATLIST.
                    CATLIST,LO=F,FN=myfile.
                    CATLIST,FN=ABC****.       <-- all files starting
                                                  with ABC


CDROP   (ICF) Abort an executing Cray job saving the output.

        Syntax:     CDROP,jsq.

        Parameters: jsq - Cray job sequence number

        See also:   CSTATUS

        Similar commands: VMS Cray Station:

        Examples:   CDROP,ABCD


CHANGE  Change some characteristics of a permanent file.

        Syntax:     CHANGE,nfn_1=ofn_1,nfn_2=ofn_2,...,nfn_n=ofn_n
                          /PW=pw,CT=ct,M=m,BR=br,PR=pr,SS=ss,NA,CE,
                           AC=ac,CP.

        Parameters: nfn_i=ofn_i - change old file name to new file
                                  name

                    PW=pw - new password
                                0 - clear the password

                    M=m   - alternate user permission mode for
                              semiprivate and public files
```

BR=br - backup requirements

br	meaning
CR	off-station backup
Y	on-station backup
MD	backed up only if on disk
N	no backup

(MD and N are not recommended)

PR=pr - preferred residence

pr	meaning
M	alternate storage - MSS
N	no preference

SS=ss - new interactive subsystem (BASIC, BATCH, EXECUTE, FORTRAN, FTNTS, NULL)

CE - clear file error code

AC=ac - may alternate users obtain information about the file? (Y or N)

CP - account number is to be replaced by the account number currently in effect

Similar commands:

Cray:	ALTACN; MODIFY
NOS/BE:	RENAME; MSCHANG
VMS:	SET PROTECTION

Examples:

```
CHANGE,mynew=myold.
          ^-- rename a file
CHANGE,myfile/CT=PU.
          ^-- make a file public
```

CHARGE Validate charging information for the job.

Syntax: CHARGE,account_number.

Parameters: account_number - your Job Order Number (10 digits or S+9 digits)

Remarks: In a batch job, the initial CHARGE statement must immediately follow the USER statement following the job statement.

The CHARGE statement may also be used to change the Job Order Number to be used for subsequent file saves.

Similar commands: Cray: ACCOUNT
NOS/BE: CHARGE
VMS: your home directory defines the job order number

Examples:

```
jobname.
USER,ABCD,batch_password.
CHARGE,1222233344.
              ^-- job charged to 1-2222-333-44
  ...
CHARGE,5666677788.
DEFINE,NEWFILE.
              ^-- file charged to 5-6666-777-88
```

CJOB (ICF) Get the status of a specific Cray job.

Syntax: `CJOB,jname,jsq,L=lfn,RT=rt.`

Parameters:

```
jname - job name from JOB statement (uppercase)

jsq   - optional job sequence number (CSTATUS)

L=    - local file to receive the status
        (default: OUTPUT)

RT=   - repeat time (seconds)
        (default: do not repeat the command)
```

See also: CSTATUS

Similar commands: VMS Cray Station:

Examples: `CJOB,MYJOB.`

CKILL (ICF) Delete an input job, kill an executing job saving only the logfile, delete an output dataset

Syntax: `CKILL,jsq.`

Parameters: jsq - job sequence number

See also: CSTATUS

Similar commands: VMS Cray Station: DROP

Examples: `CKILL,abcd.`

CKP Take a checkpoint dump.

Syntax: `CKP,lfn_1,lfn_2,...,lfn_n.`

Parameters: lfn_i - a file to be included in the checkpoint dump
(Default: all local files)

Similar commands: NOS/BE: CKP

Examples: `CKP.`

CLEAR Release all (or all but one or more specified) files assigned to the job.

Syntax:

```
CLEAR.                          <-- all files

CLEAR,*,lfn1,lfn2,...,lfnn.     <-- all except those
                                    named
```

Remarks: Checkpoint and no-auto-drop files are not released.

See also: RETURN, SETFS

Similar commands: NOS/BE: CLEAR

Examples:

```
CLEAR.
CLEAR,*,keepfyl.
```

COBOL5 Compile COBOL 74 program.

Syntax: `COBOL5,B=b,I=i,L=l,LO=lo,PD=pd,SY.`

Parameters:

B=PUNCHB	Produce punched binary decks of all routines
B=lfn	Put binary into a file
B	Same as B=BIN
B=0	No binary output
omitted	Same as B=LGO
I=lfn	FORTRAN source input is in lfn
I	Same as I=COMPILE
omitted	Same as I=INPUT
L=lfn	Output lists to file lfn
L=0	Listings are suppressed
L	Same as L=LIST
omitted	Same as L=OUTPUT

LO=op/op/... Listing options (see L parameter)

op	meaning
M	address map
O	object code listing (use only if requested by Code 1893)
R	cross-reference map
S	source code list

LO Same as LO=M/S/R
LO=0 No M, O, R, S information
omitted Same as LO=S

PD=8 Print density (listings at 8 lines per inch, single spacing)
PD=6 listings at 6 lpi single spacing
PD=4 listings at 8 lpi double spacing
PD=3 listings at 6 lpi double spacing
PD Same as PD=8
omitted Same as PD=6

SY Syntax check only; do not produce object code
(cuts compilation time roughly in half)

Similar commands: NOS/BE: COBOL5
VMS: COBOL

Examples:

```
COBOL5.      Defaults to: I=INPUT,L=OUTPUT,LO=S,B=LGO

COBOL5,I=INP,L=OUTP,LO,SY,PD=8.
```

COMMENT Place a comment in the system dayfile and the dayfile for any of your jobs.

Syntax:

```
COMMENT,jsn.comment
COMMENT.comment
*comment
```

Parameters: jsn - jsn of job to receive the comment
Default: the current job

comment - the message to be put into the dayfile

See also: NOTE (for on-line messages from a procedure)

Similar commands: Cray: *
NOS/BE: COMMENT
VMS: !

Examples:

```
COMMENT.MARK 1      <-- these are
*MARK 1             <--  the same
```

COPY Copy data from one file to another.

Syntax:

```
COPY,I=lfni,O=lfno,V=x,M=c,TC=tc,N=copycnt,
     BS=bsize,CC=charcnt,EL=erlimit,
     PO=plp2...pn,L=lfnl,NS=ns.

COPY,lfni,lfno,x,c,tc,copycnt,bsize,charcnt,
     erlimit,plp2...pn,lfnl,ns.
```

Parameters:

I= - file to be copied
(default: INPUT)

O= - output file
(default: OUTPUT)

V= - non-null to rewind, copy, rewind, verify, rewind both files (x must not be 0)
(default: no verify)

M= - coded files

c	meaning
C1	only input is coded
C2	only output is coded
x	both input and output are coded (x is non-null, except C1, C2)

(default: binary)

TC= - termination condition with N=

tc	meaning
F or EOF	N is number of files
I or EOI	N is ignored (copy to end-of-information)
D or EOD	N is number of double EOFs to copy to (N>1,TC=D,VERIFY verifies only to the first empty file)

(default: TC=D)

N= - copy count (meaning determined by TC=)
(default: N=1)

BS= - maximum block size (in CM words) of S or L tape PRU (cannot be used with CC=)
(defaults: S tape: 1000 octal;
L tape: 2000 octal)

CC= - maximum number of characters in S or L tape block
(default: not used; size from BS=)

EL= - number of non-fatal errors before abort;
EL=U for unlimited error processing
(default: 0)

PO= - processing options:

pi	meaning
E	copy input blocks with parity or block-too-long errors (default: error blocks skipped)
D	delete noise blocks in copy from disk, I- or SI-tape to S- or L-tape (defaults: binary padded with 00B to noise block size; coded padded with blanks)
R	allow record splitting in copy from disk, I- or SI-tape to S- or L-tape (default: splitting not allowed)
M	copy according to TC; do not write EOFs (default: write EOF after each *file*)

L= - alternate file for parity error messages for EL<>0; cannot be same as I=, O=
(default: OUTPUT)

NS= - noise size for S, L or F input tapes
(maximum: 41; NS=0 uses default of 18)

Similar commands:

Cray:	COPYD; COPYF; COPYR; COPYU
NOS/BE:	COPY; COPYBF; COPYCF; COPYE; COPYF; COPYRM; COPYSF; (last 4 DTRC)
VMS:	COPY

Examples:

```
COPY,a,b.
COPY,a,b,verify.
```

COPYBF Copy a multi-file file in binary mode.
COPYBR Copy records from one file to another in binary mode.

Syntax:

```
COPYBF,lfn_in,lfn_out,nfiles,c.
COPYBR,lfn_in,lfn_out,nrecs,c.
```

Parameters: nfiles - decimal number of files to copy

nrecs - decimal number of records to copy

c - non-null indicates coded S or L tape

Defaults:

```
COPYBF,INPUT,OUTPUT,1.
COPYBR,INPUT,OUTPUT,1.
```

Remarks: Not recommended for S or L tapes.

Copies from current position.

If lfn_in=lfn_out, the file is read.

See also: COPYCF, COPYCR, COPYEI

Similar commands: Cray: COPYD; COPYF; COPYU
NOS/BE: COPYBF; COPYBR

Examples:

```
COPYBF,fyl1,fyl2,4.
COPYBR,fyl1,fyl2,125.
```

COPYCF Copy a coded multi-file file.
COPYCR Copy records from one coded file to another.

Syntax:

```
COPYCF,lfn_in,lfn_out,nfiles,
       fchar,lchar,na.                <-- 6-bit

COPYCR,lfn_in,lfn_out,nrecs,
       fchar,lchar,na.                <-- 6-bit

COPYCF,lfn_in,lfn_out,nfiles,,,na.    <-- 6/12-bit

COPYCR,lfn_in,lfn_out,nrecs,,,na.     <-- 6/12-bit
```

Parameters: nfiles - decimal number of files to copy

nrecs - decimal number of records to copy

fchar - first character position of line to copy

lchar - last character position of line to copy

na - non-null to not abort if no line terminator before EOR

Defaults:

```
COPYCF,INPUT,OUTPUT,1,1,136.
COPYCR,INPUT,OUTPUT,1,1,136.
```

Remarks: Not recommended for S or L tapes.

Copies from current position.

If lfn_in=lfn_out, the file is read.

A coded file contains lines of 500 or fewer characters, terminated with a zero-byte (12-bits).

Lines longer than 500 6-bit characters are truncated.

See also: COPYBF, COPYBR, COPYEI

Similar commands: Cray: COPYD; COPYF; COPYU
NOS/BE: COPYCF, COPYCR

Examples:

```
COPYCF,cfyl1,cfyl2.
COPYCR,cfyl1,cfyl2,2,7,35,x.
```

COPYEI Copy one file to another.

Syntax: COPYEI,lfn_in,lfn_out,x,c.

Parameters: lfn_in - file to be copied

lfn_out - the copy of the input file

x - non-null to rewind, copy, rewind, verify, rewind both files

c - non-null indicates coded S or L tapes (default: binary)

Defaults: COPYEI,INPUT,OUTPUT.

Remarks: Not recommended for S or L tapes.

Copies from current position.

If lfn_in=lfn_out, the file is read to end-of-information.

See also: COPY, COPYBF, COPYCF

Similar commands: Cray: COPYD
NOS/BE: COPYE
VMS: COPY

Examples:

```
COPYEI,myin,myout.
```

COPYL Selective single replacement of object modules.

Syntax: COPYL,oldfyl,repfyl,newfyl,last,flag.

Parameters:

oldfyl - old master binary file of object modules
(Default: OLD)

repfyl - replacement file of object modules
(Default: LGO)

newfyl - new master binary file of object modules
(Default: NEW)

last - name of last record in oldfyl to be processed
(Default: all records processed from current position to EOF or EOI)

flag - processing options

- R - rewind oldfyl and newfyl before processing
(repfyl is always rewound)
- A - append to newfyl all records in repfyl which do not match any of oldfyl
(Default: non-matching records ignored)
- T - match record name but not type
(Default: must match record name and type)
- E - process oldfyl through EOI

(may be selected in any combination: RA, ART, TEAR, etc., up to four characters; extra characters ignored)
(default: option not selected)

Defaults: COPYL,OLD,LGO,NEW.

Remarks: Oldfyl is processed forward only, but binary will be searched as often as needed.

See also: COPYLM

Similar commands:

Cray:	BUILD
NOS/BE:	COPYL
VMS:	LIBRARIAN

Examples: COPYL,oldlgo,lgo,newlgo.

COPYLM Selective multiple replacement of object modules.

Syntax:
`COPYLM,oldfyl,binary,newfyl.`
`COPYLM,oldfyl,binary,newfyl,last,flag.`

Parameters: see COPYL; flags T, E do not apply to COPYLM

Defaults: `COPYLM,OLD,LGO,NEW.`

Remarks: All occurrences of a module in oldfyl will be replaced by the first occurrence of a module with the same name in binary.

See also: COPYL

Similar commands:

Cray:	BUILD
NOS/BE:	COPYLM
VMS:	LIBRARIAN

Examples: `COPYLM,oldlgo,lgo,newlgo,R.`

COPYSBF Copy a file, shifting the lines one character to the right for printing on a printer.

Syntax: `COPYSBF,lfn_in,lfn_out,nfiles,na.`

Parameters:

lfn_in - file to be copied

lfn_out - the copy of the input file

nfiles - decimal number of files to copy

na - non-null to not abort if no line terminator before EOR

Defaults: `COPYSBF,INPUT,OUTPUT,1.`

Remarks: Not recommended for S or L tapes.

Copies from current position.

If lfn_in=lfn_out, the file is read.

A coded file contains lines of 500 or fewer characters, terminated with a zero-byte (12-bits).

Lines longer than 500 6-bit characters are truncated.

A page eject is inserted at the start of each logical record.

Similar commands:
Cray: COPYD
NOS/BE: COPYSBF; COPYSF (DTRC); COPYSR (DTRC)
VMS: VSYS:CPYSF (DTRC)

Examples: `COPYSBF,myprog.`

COPYX Copy a file until a user-specified condition is met.

Syntax: `COPYX,lfn_in,lfn_out,x,b,c.`

Parameters: lfn_in - file to be copied

lfn_out - the copy of the input file

x - copy specifications

x	meaning
n	decimal number of records
00	copy through first zero-length record
name	copy through this record
type/name	copy through this record

b - backspace control

x	meaning
0	no backspace
1	backspace input file one record after copy
2	backspace output file one record after copy
3	backspace both files one record after copy

(ignored if EOF or EOI before x met)

c - non-null indicates coded S or L tape (default: binary)

Defaults: `COPYSBF,INPUT,OUTPUT,1.`

Remarks: Not recommended for S or L tapes.

Copies from current position.

If lfn_in=lfn_out, the file is read.

A coded file contains lines of 500 or fewer characters, terminated with a zero-byte (12-bits).

Lines longer than 500 6-bit characters are truncated.

A page eject is inserted at the start of each logical record.

See also: COPYBR, COPYCF, COPYCR

Similar commands: Cray: COPYF; COPYR; COPYU
NOS/BE: COPYCF

Examples: COPYX,fyl1,fyl2,125.

CRERUN (ICF) Immediately end processing of specified job, delete output, and resubmit to input queue, if allowed.

Syntax: CRERUN,jsq.

Parameters: jsq - job sequence number

Remarks: A job cannot be rerun if:

- a dataset has been adjusted, modified, saved, deleted, or written on
- RERUN,DISABLE has been executed

See also: RERUN

Examples: CRERUN,9876.

CSET (IAF) Change the terminal's character set mode.

Syntax: CSET,mode

Parameters: mode - one of:
ASCII - ASCII 128-character set
NORMAL - ASCII graphic 64-character set

Remarks: CSET may appear in a procedure.

See also: CSET,NORMAL does not affect AUTO or BRIEF mode.

Examples: CSET,ASCII

CSTATUS (ICF) Get the status of jobs, and input and output datasets.

Syntax: CSTATUS,queues,ST=start,L=lfn,RT=rt.

Parameters: queues - one or more of:

value	meaning
E or EXECUTION	execution queue
I or INPUT	input queue
O or OUTPUT	output queue
R or RECEIVING	Cray mainframe receiving queue
S or SENDING	Cray mainframe sending queue
A or ALL	all of the above

(Default: ALL)

ST= - decimal number of entries to skip before starting the display (default: 0)

L= - local file to receive the status (default: OUTPUT)

RT= - repeat time (seconds) (default: do not repeat the command)

Remarks: If RT is specified, %2 may be needed to cancel the output.

Examples: CSTATUS.

CSUBMIT Submit a job to a Cray mainframe.

Syntax: CSUBMIT,lfn,RB=user,NO,TO.

Parameters: lfn - local file containing the Cray job (display code or 8/12-bit ASCII)

RB= - (remote batch submission) user to receive the output

RB - put the output into the print queue for you (default: print at Centrel Site)

NO - drop output at job termination

TO - put the output in the wait queue

Remarks: Other parameters are available for running another user's job.

```
Similar commands:  VMS Cray Station:  SUBMIT
                   VMS:               CRAY SUBMIT

Examples:     CSUBMIT,mycray.
                           ^-- output to 860 printer

              CSUBMIT,mycray,RB=un.
                           ^-- output to un's output queue
                               (see QGET)

              CSUBMIT,mycray,RB.
                           ^-- output to your output queue
                               (see QGET)
```

CTASK Transfers the file between the Cray and the CDC NOS front-end.

```
Syntax:       CTASK,ALL,code.

Parameters:   ALL  - include the dayfile from the CDC NOS job
                     with the Cray logfile
                     (default: the dayfile is not included)

              code - internal code of the CDC NOS file
                       ASCII8 - 8/12-bit ASCII
                       DIS    - display code
                     (default: display code, unless the whole
                               job is ASCII8)
```

Remarks: The text field of a Cray ACQUIRE, DISPOSE or FETCH statement can include NOS commands to fetch or store the file. CTASK causes the file transfer to occur.

Examples: See pages 2-1-8, 3-1-3.

CTIME Put the accumulated CPU time (in seconds) into the job's dayfile.

```
Syntax:       CTIME.

See also:     RTIME, STIME

Similar commands:  Cray:
                   NOS/BE:  PTIM (DTRC); SUMMARY
                   VMS:     ^T

Examples:     CTIME.
```

DAYFILE Write the job's dayfile (or a subset) to a file.

Syntax:

```
DAYFILE,L=lfn,FR=string,OP=op,PD=pd,PL=pl,
        I=infile.
DAYFILE,lfn,string,op,pd,pl,infile.
```

Parameters:

L= - the file to which the dayfile is to be written (on a new page if OUTPUT or if PD or PL specified)
(Default: L=OUTPUT)

FR= - search string in field OP for starting the copy. $-delimited if any non-alphanumerics in the string (time starts with a blank; interactive commands start with a "$"; for example: $$$OLD$). If found, the dayfile is copied from this point. If not found, a message (connected or disconnected L) and the entire dayfile (connected L) are written.

OP= - search option

op	meaning
T	search time field
M	search message field
I	incremental dump (from point of last DAYFILE command)
F	full dump

(Defaults: OP=M (FR, but no OP);
OP=F (L disconnected);
OP=I (L connected)

PD= - print density

pd	meaning
3	double space; 6 lpi
4	double space; 8 lpi
6	single space; 6 lpi
8	single space; 8 lpi

PL= - page size

pd	page size	defaults
3	pl / 2	30 lines
4	pl / 2	30 lines
6	pl	60 lines
8	pl	60 lines

I= - file containing a dayfile for input
(Default: the active dayfile)

Remarks: A paginated dayfile cannot be used.

Similar commands: Cray: ECHO
VMS: /LOG qualifier

Examples:

```
DAYFILE,,$ 11.21.$,T.
            ^-- start with the last occurrence of
                11.21. in the time field

DAYFILE,I=DAY,FR=$$$GET,STATS.$.
            ^-- start with the last occurrence of
                $GET,STATS. in the message field
                of the dayfile in file DAY
```

DEFINE Create an empty direct permanent file.

Syntax:

```
DEFINE,lfn_1=pfn_1,lfn_2=pfn_2,...,lfn_n=pfn_n
       /PW=pw,CT=ct,M=m,BR=br,PR=pr,S=space,NA,
       AC=ac.
```

Parameters: See CHANGE.

PW=pw - a 1- to 7-character password required by others for access

BR=br - backup requirements

br	meaning
CR	off-station backup
Y	on-station backup
MD	backed up only if on disk
N	no backup

(MD and N are not recommended)
(Default: Y)

CT=ct - (Default: CT=PRIVATE)

PR=pr - preferred file residence

pr	meaning
D	disk
L	locked to disk
M	alternate storage (MSS)
N	no preference
T	tape alternate storage

(Default: PR=N)

S=space - number of PRUs requested for the file
(default: the minimum number of blocks
(a multiple of 704 PRUs)
needed to hold the file)

Similar commands: Cray: SAVE
NOS/BE: REQUEST,REWIND,CATALOG
VMS: CREATE

Examples:

```
DEFINE,myfile/CT=PU.
```

DIAL (IAF) Send a one-line message to another user.

Syntax:

```
DIAL,jsn,sss
```

Parameters: jsn - job sequence number of the receiving terminal

sss - the one-line message

Remarks: You must be in the ACCESS subsystem.

No queuing takes place if jsn is busy.

See also: WHATJSN

Similar commands: NOS/BE: SEND
VMS: PHONE

Examples:

```
ACCESS
WHATJSN
DIAL,jsn,message
```

DISPLAY Evaluate an expression and put the result into the job's dayfile in octal and decimal.

Syntax:

```
DISPLAY,exp1,exp2,...,expn.
```

Parameters: expi - any valid symbolic name or expression

Remarks: The largest value which can be displayed is 10 digits.

See also: SET

Similar commands: Cray: PRINT
NOS/BE: DISPLAY
VMS: WRITE SYS$OUTPUT

Examples:

```
1) DISPLAY,TIME.
         1253    2345B     <-- if time is 12:53
```

```
2) SET,R1=99.
   DISPLAY,R1.              <-- register R1
        99    143B

3) DISPLAY,143B.            <-- octal number with B
        99    143B

4) DISPLAY,3/2.             <-- integer division
        1     1B
```

DMB Binary dump of exchange package.

Syntax: `DMB,ordinal.`

Parameters: ordinal - an octal number (0-777777) used to create the dump record number (D plus ordinal) - ordinal > 377777 aborts the job after the dump
(Default: 0 ==> D000000)

Remarks: The dump is written to file ZZZZDMB (an unconnected local file), which is never rewound.

See also: DMD, DMP

Similar commands: Cray: DUMPJOB,DUMP
NOS/BE: DMP

Examples: `DMB.`

DMD Dump the exchange package or central memory in both octal and display code.

Syntax:

```
DMD,fwa,lwa.          (1)
DMD,lwa.              (2)
DMD.                  (3)
```

Parameters: fwa - first word address of memory to be dumped (relative to RA)

lwa - last word address to be dumped (relative to RA)

Format 1: dump a specified range of memory
Format 2: dump from RA+0 thru specified lwa
Format 3: dump the exchange package and 40B locations before and after the program address

Remarks: The dump is written four words per line to file OUTPUT. Interactively, it is generally written to file ZZZDUMP, which is never rewound.

See also: DMP

Similar commands:
Cray: DUMPJOB,DUMP
NOS/BE: DMD
VMS:

Examples:

```
DMD,50000,60000.
```

DMP Dump the exchange package or central memory in octal.

Syntax: Same as DMD.

Similar commands:
Cray: DUMPJOB/DUMP
NOS/BE: DMP
VMS:

Examples:

```
DMP,47000.
```

DROP Drop any of your executing or queued files (except the job issuing the DROP command).

Syntax:

```
DROP,JSN=jsn,DC=q,UJN=ujn,OP=R.
DROP,jsn,q,ujn,R.
```

Parameters: JSN= - either or both
UJN= - may be specified

DC= - disposition code

dc	meaning
WT	waiting jobs
PR	print jobs
PU	punch jobs
PL	plot jobs
IN	input jobs
EX	executing jobs
ALL	all your jobs

(Defaults: none (JSN=,UJN= omitted);
ALL (JSN= or UJN= specified))

OP= - drop executing jobs without EXIT, but with single-reprieve processing

```
Similar commands:  NOS/BE:  DROP; EVICT; KILL
                   VMS:     STOP
```

```
Examples:    DROP,ABCD.       <-- drop executing job ABCD
             DROP,JSN=ABCD,OP=R.
                              ^-- drop executing job with
                                  single-reprieve but no EXIT
             DROP,,PR.        <-- drop all print jobs
             DROP.            <-- invalid (JSN, UJN or DC
                                  required)
```

ELSE Terminate skipping (false IF command with same label), or initiate skipping (true IF command with same label) to ENDIF with same label.

Syntax: ELSE,label.

Parameters: label - alphanumeric string (axxxxxxxxx, 1-10 characters)

See also: IF

```
Similar commands:  Cray:    ELSE; ELSEIF
                   NOS/BE:  ELSE
                   VMS:     ELSE; ELSEIF (VMS 5.0)
```

```
Examples:    SET,R1=1.
               ...

             IF,R1=1,DOIT.
               <statements to do if true>
             ELSE,DOIT.
               <statements to do if false>
             ENDIF,DOIT.
```

ENDIF Terminate skipping by a SKIP, IF, or ELSE command with a matching label.

Syntax: ENDIF,label.

Parameters: label - alphanumeric string (1-10 characters, starting with a letter)

See also: ELSE, IF

```
Similar commands:  Cray, NOS/BE:  ENDIF
                   VMS:           ENDIF (VMS 5.0)
```

Examples: See IF.

ENDW The end of a WHILE loop.

Syntax: `ENDW,label.`

Parameters: label - alphanumeric string (1-10 characters, starting with a letter)

See also: WHILE

Similar commands: Cray: ENDLOOP
NOS/BE: ENDW

Examples:

```
WHILE,R1<5,DOIT.
  ...
SET,R1=R1+1.
ENDW,DOIT.
```

ENQUIRE Get information about your jobs.

Syntax:

```
ENQUIRE,OP=p1p2...pn,FN=lfn1,O=lfn2.
ENQUIRE,p1p2...pn.
ENQUIRE,JSN=jsn,O=lfn2.
ENQUIRE,UJN=ujn,O=lfn2.
ENQUIRE.
```

Parameters: OP= - Up to 7 of the following options:

pi	meaning
A	same as BDRUJLF
B	identification and priority info
D	resources demanded and assigned
F	status of files assigned to your job
J	contents of control registers, error flags fields, succeeding commands
L	loader info including status of CID
R	amount of resources used (CPU time, mass storage, perm file, and adder activity, SRUs used)
S	accumulated SRUs
T	accumulated cpu time
U	initial amount of resources available (seconds, job step RSU, account block SRU, remaining resources available for dayfile messages, commands, and mass storage)

(Default: OP=A)

JSN= - returns detailed report on this job

UJN= - returns one-line report for each of your jobs

```
                    FN=  - returns the status of local file lfnl
                    O=   - writes the output to file lfn2
                    (Default: interactive: a 2-line report on the
                          current job)

     Similar commands:  NOS/BE:  ASSETS, FILES, FIND, J, MYQ, Q,
                                 SUMMARY
                        VMS:     SHOW SYSTEM

     Examples:      /ENQUIRE,B       <-- display system activity
                    /ENQUIRE,JSN     <-- display all of your jsn's
                    /ENQUIRE,JSN=abcd,O=out
                                     ^-- display status and remaining
                                         commands for job ABCD; write
                                         display into file OUT
                    /ENQUIRE         <-- a 2-line report on the
                                         current job
                    /ENQUIRE,UJN     <-- display status of your jobs
```

ENTER   Enter a series of commands on one line.

```
     Syntax:        ENTER./command1/command2/.../commandn

     Parameters:    /        - delimiter - any character not in any
                               commandi - immediately follows the
                               terminator

                    commandi - any NOS command (except interactive
                               commands with no batch counterpart)

     Remarks:       The system supplies a terminator if it is missing
                    from any commandi.

     Examples:      BATCH            <-- enter the batch subsystem
                    $RFL,0.          <-- displayed on entry to batch
                    /ENTER.\get,fprog\ftn5,i=fprog\map,part\lgo\
                           \exit\dmp\rewind,zzzdump\copy,zzzdump
                                     ^-- (must fit on one line)
                                         compile and execute a program
```

ERRMSG   Control the display of error messages in a procedure.

```
     Syntax:        ERRMSG,status.

     Parameters:    status - OFF - turn off display of messages
                             ON  - turn on display of messages
                             (Default: ON)

     Remarks:       ERRMSG has no effect in a batch job.
```

Similar commands: NOS/BE: DAY/DAYFILE

Examples: ERRMSG,OFF.

EVICT Release file space but not the file assignment.

Syntax: EVICT,lfn1,lfn2,....

Remarks: If the file is a magnetic tape or a read-only disk file, the file assignment is also released.

Similar commands: NOS/BE: REWIND,ALTER (disk)
VMS: create a new version (disk)

Examples: EVICT,myfyl.

EXECUTE (IAF) Select the execute subsystem.

Remarks: Not recommended at DTRC.

EXECUTE (Loader) Complete loading, generate load map (if requested), begin execution; or execute at a specific entry point.

Syntax: EXECUTE.

EXECUTE,pname,plist.

Parameters: pname - a specific entry point at which to begin execution

plist - list of parameters

See also: LGO, name

Similar commands: NOS/BE: EXECUTE

Examples: LOAD,lgo.
EXECUTE.

EXIT Resume processing commands after a previous error.

Syntax: EXIT.

Remarks: When used in a procedure, precede it with a SKIP or REVERT, because EXIT terminates the current and all calling procedures without restoring the registers.

See also: NOEXIT, ONEXIT

Similar commands:		
	Cray, NOS/BE:	EXIT
	VMS:	ON condition

Examples:

```
FTN5.
LGO.
EXIT.
COMMENT. Program failed.
```

EXPLAIN (IAF) Retrieve an on-line version of a CDC manual.

Syntax:

```
EXPLAIN,M=manual

EXPLAIN,manual
```

Parameters: manual - the desired manual

See also: HELP, HELPME

Similar commands:		
	NOS/BE:	BEGIN,DOCGET; interactive procedures
	VMS:	HELP
	VMS Cray Station:	HELP

Examples:

```
EXPLAIN.
```

FCOPY Convert a file from one character set to another.

Syntax:

```
FCOPY,P=lfn,N=lfn,PC=cs,NC=cs,PL=lt,NL=lt,
      FL=fl,LB=lb,R,A.
```

Parameters:

P= - input file to be converted
(default: OLD)

N= - output converted file
(default: NEW)

PC= - character set of input file

cs	meaning
ASCII	6/12-bit display code supporting ASCII 63- or 64-char on current system
ASCII8	7-bit ASCII, rt-just in 12 bits
ASCII88	8-bit ASCII, rt-just in 8 bits
ASCII63	6/12-bit display code supporting ASCII 63-character set

ASCII64	6/12-bit display code supporting ASCII 64-character set
ASCFL	8-bit ASCII on S tapes (fixed line length, no line terminators)
DIS	6-bit display code supporting CDC 63- or 64-char on current system
DIS63	6-bit display code supporting CDC 63-character set
DIS64	6-bit display code supporting CDC 64-character set
EBCFL	8-bit EBCDIC on S tapes (fixed line length, no line terminators)

(default: ASCII)

NC= - output character set
(default: ASCII8)

PL= - input line terminator

value	meaning
ZB	zero byte
CR	carriage return
FF	form feed
LF	line feed
US	unit separator
RS	record separator
CRLF	carriage return and line feed
LFCR	line feed and carriage return
n	specified octal character

(defaults: most: ZB;
ASCII88: US;
ASCFL, EBCFL: no terminators)

NL= - output line terminator
(defaults: same as for PL=)

FL= - length of fixed length lines for S tapes
(default: 80; valid only for ASCFL, EBCFL)

LB= - number of lines per block
(default: 3840/fl; valid only for ASCFL, EBCFL)

R - rewind input and output files before and after processing
(default: no rewind)

A - abort on errors
(default: no abort)

Remarks: Maximum line length is 160 (12-bit codes) or 320 (6-bit codes). Longer lines are truncated.

Files converted to 7-bit ASCII can be printed on a Central Site printer and on some remote batch printers. They cannot be listed at an interactive printer.

```
Examples:     FCOPY,P=my63,PC=DIS63,N=my64,NC=DIS64,R.
                                ^-- convert NOS/BE 63-character set
                                    file to NOS 64-character set
              FCOPY,P=mya,N=mya8.
                                ^-- convert 6/12-bit ASCII to 8/12-
                                    bit ASCII suitable for ROUTE-ing
                                    with EC=A9 to a printer;
                                    for input to Cray;
                                    for FSE directives in batch
              FCOPY,P=tape,N=disk,PC=ASCFL,NC=DIS,FL=120,LB=30.
                                ^-- convert ASCII foreign tape of
                                    120-character records blocked 30
                                    to an internal display code file
```

FILE (CRM) Describe the attributes of a file.

```
Syntax:       FILE,lfn,keys.
              FILE,lfn=xxxxxxx,keys.

Parameters:   lfn      - file to be described
              =xxxxxxx - new name for file
              keys     - keyword parameters for the various
                         attributes and their values -- some
                         are:
                          ASCII= - for interactive terminals
                                     0 - 64-char display code
                                     1 - 95-char ASCII
                                     2 - 128-char ASCII
                                   (default: ASCII=0)
                          BFS=   - buffer size
                                     0 - system provides space
                                     n - octal buffer size
                                   (default: BFS=0)
                          BT=    - block type
                                     I - internal
                                     C - character count
                                     K - record count
                                     E - exact record count
                                   (default: BT=I)
                          CF=    - close file action
                                     N - no rewind
                                     R - rewind
                                     U - rewind and unload
                                   (default: CF=N)
```

```
CM=     - conversion mode
           NO  - no conversion
           YES - convert external to
                 internal
          (default: CM=NO)
CNF=    - connect file flag
           NO  - disk or tape file
           YES - terminal file
          (default: CNF=NO)
DFC=    - dayfile control
           0 - fatal errors to dayfile
           1 - errors to dayfile
           2 - notes to dayfile
           3 - errors and notes
          (default: DFC=0)
EFC=    - error file control
          (same as DFC, except errors
          written to error file
          ZZZZZEG)
EO=     - parity error processing
           T  - terminate with fatal
                error
           D  - drop bad data
           A  - accept bad data
           TD - same as T, D, A plus
           DD   display the error
           AD   block on error file
                ZZZZZEG
          (default: EO=T)
ERL=    - trivial error count
           0 - no limit
           n - number of trivial
               errors to accept
               (max: 551)
          (default: ERL=0)
FF=     - flush sequential files on
          abnormal termination
           NO  - buffers not flushed
           YES - output scratch file
                 buffers flushed
          (default: FF=NO)
FL=     - fixed length (RT=T) or full
          length (RT=Z)
           0 - must be defined for
               open
           n - decimal length
               RT=F: n is 10-1310710
               RT=Z: n is  1-1310710
          (default: FL=0)
```

```
FO=     - file organization
           SQ - sequential
           WA - word addressable
          (default: FO=SQ)
LCR=    - label creation flag
           CRT - create new label
           CHK - check existing label
          (default: LCR=CRT)
LFN=    - new local file name
LT=     - label type
           UL  - unlabelled
           S   - ANSI standard
           NS  - nonstandard
           ANY - any label type (no
                 user processing)
          (default: LT=UL)
MBL=    - maximum block length (in
          characters)
           0 - BT=I  - 5120
               BT=C  - 5120  (S tapes)
                       BFS-20(L tapes)
               other - error
           n - length
                BT=K,E,RT=Z - >= FL+10
                BT=I        - MBL
          (default: MBL=0)
MRL=    - maximum record length
           0 - no maximum
           n - maximum number of
               characters
               (max: 1310710)
OF=     - open file action
           N - no rewind
           R - rewind
          (default: OF=N)
PD=     - I/O processing
           INPUT  - read
           OUTPUT - write
           IO     - read/write
          (default: PD=INPUT)
RB=     - records per block for BT=K
           0 - same as RB=1
           n - number (max: 4095)
          (default: RB=1)
```

```
                    RT=    - record type
                              W - control word
                              F - fixed length
                              R - record mark
                              Z - zero-byte terminated
                              D - decimal character count
                              T - trailer count
                              U - undefined
                              S - system-logical-records
                             (default: RT=W)

Remarks:     Other parameters include: BBH, B8F, CL, CP, C1,
             HL, LBL, LL, LP, MFN, MNB, MNF, MUL, OMIT, PC,
             PNO, REL, RMK, SB, SBF, SPR, TL, ULP, USE, VF.

Similar commands:  NOS/BE:  FILE

Examples:    FILE,PRT,BT=C,RT=Z,MRL=150.
                      ^-- zero-byte terminated print file

             FILE,STRANGR,BT=K,RT=F,RB=10,MRL=80,MBL=800,EO=A,
                         ERL=25,BFS=512,CM=YES.
                      ^-- stranger blocked coded tape

             FILE,INPUT,LFN=DATA.
                      ^-- substitute alternate input file
```

```
FORM    File Organization and Record Manager.

        Syntax:      FORM,I=dirfyl,B=owncode.

        Parameters:  I= - directive file
                          (default: INPUT)

                     B= - owncode routines to be loaded
                          (default: B=0)

        Directives:  Parameters for sequential files are shown
                     (additional parameters are available):

                     Input file:  INP(lfn,POS=+-n,MAX=n,REW=r)

                       lfn - the input file
                       POS= - skip n logical records forward or
                              backward before processing
                              (range: 1-16383; default: no limit)
                       MAX= - maximum number of records to be processed
                              (range: 1-8,388,607; default: no limit)
```

```
  REW= - rewind at end
           N - no rewind
           R - rewind
           U - rewind and unload (tape)
         (default: R; ignored for INPUT)

Output file (up to 20):  OUT(lfn,POS=+-n,MAX=n,
                             REW=r,BGD=g,SEL=a)

  lfn - the output file (up to 20 OUT files may
        be specified)
        (default: OUTPUT)
  POS= - same as for INP
  MAX= - same as for INP
  REW= - same as for INP
         (default: R; ignored for OUTPUT, PUNCH,
          PUNCHB)
  BGD= - preset output record
           X - blank (55B)
           N - display code zero (33B)
           B - binary zero
           E - floating point zero
           C - same as input record
         (default: B)
  SEL= - selection criteria
           ALL - copy all records with QAL
                 processing as requested
           QRO - copy only records meeting QAL
                 criteria
         (default: SEL=QRO)

Non-standard label:  NON(lfn,ORD=n,LEN=n,LBL=lit)

Record qualification:  QAL(lfn,condition)

Record reformatting:  REF(lfn,entry,entry,...)

  entry - out_iTm=in_iTm  - move input field to
                            output field

          out_iTm=literal - put literal into
                            output field of all
                            records

          iTm - field specification
                  i - initial position (decimal)
                  T - field type
                        X - character
                       (also: E, D, U, I, S, N, Z)
                  m - length

          literal - dollar-delimited ($...$)
```

```
Print:  PRT(lfn,FMT=f,PGL=n,TOP=n,TTL=lit)

  lfn  - the output file
  FMT= - line spacing
           1 - single space
           2 - double space
           3 - triple space
           A - first character of line is
               carriage control
           D - dump the lines single spaced, 100-
               character lines
  PGL= - number of print lines (including the
         title line) per page
         (max: 60 (f=1), 30 (f=2), 20 (f=3);
          default: 60/f)
  TOP= - line of page for the title
         (range: 2-60; defaults: 1 (TTL omitted),
                                 2 (TTL given))
  TTL= - the page title (up to 136 characters)
         (ignored for f=D)

IBM S/360 magnetic tape conversion:
      CON(lfn,RID=lit,descr...,RID=lit,descr...)

Execute:  XEQ(ERR=e,COL=lit,...,FIN)

  ERR= - error processing for unrecovered tape
         parity errors:
           ASV - abandon FORM run
           ANO - abandon FORM run; get rid of
                 output disk files, OUTPUT and
                 any partially written tapes
                 remain
           CSV - continue, accept bad block,
                 dump bad block to ZZZZZEF
           CNO - continue, delete bad block,
                 dump bad block to ZZZZZEF
  COL= - alternate collating sequence
         (up to 64 characters; those not
         specified collate equal and higher than
         the highest specified)
```

Remarks: FORM has many functions including reblocking sequential files.

FORM may also be called from a Fortran or Cobol program.

FILE statements are used to describe the file blocking.

See also: FCOPY (much easier to use)

Similar commands: NOS/BE: COPYRM; FORM

FORTRAN (IAF) Select the FORTRAN subsystem.

Remarks: Use FSE and FTN5.

FSE Invoke the full screen editor.

Syntax:

```
FSE,FN=file,OP=access,I=input,L=output,
    IP=procedure,WF=workfile.directives

FSE,file,access,input,output,procedure,
    workfile.directives
```

Parameters:

file - local or permanent file to be edited
(Default: most recently edited file during job)

access - character set code and file location

access	abb	meaning
DISPLAY or NORMAL	D N	6-bit display code (default if your terminal is in NORMAL mode)
ASCII	A	6/12-bit display code (default if your terminal is in ASCII mode)
ASCII8	8	7-bit ASCII right-justified in 12 bits)
GET	G	access an existing file (may be used with above)

input - input directive file
(Default: INPUT)

output - output listing file
(Default: OUTPUT)

procedure - alternate FSE procedure library
(Default: FSEPROC)

workfile	- alternate FSE work file (Default: ZZZWORK)
directives	- initial directives (use ";" to separate directives)

See also: DTRC/CMLD-88/15, "CDC NOS Full Screen Editor (FSE) User's Guide"

Similar commands:	Cray:	TEDI
	NOS/BE:	NETED; EDITOR
	VMS:	EDIT/EDT; EDIT/TPU (EVE)

Examples:	FSE,myfile,G.	<-- GET/ATTACH existing file to edit
	FSE.	<-- resume previous editing session

FTN5 Compile Fortran 77 program.

Syntax:

```
FTN5,ANSI=ansi,B=b,BL=bl,CS=cs,DB=db,DO=do,DS=ds,
     E=e,EL=el,ET=et,GO=go,I=i,L=l,LO=lo,MD=md,
     OPT=opt,PD=pd,PL=pl,PN=pn,PS=ps,PW=pw,QC=qc,
     REW=rew,ROUND=round,SEQ=seq.
```

Parameters:	ANSI=T	Flag Non-ANSI (trivial)
	ANSI=F	Flag Non-ANSI (fatal)
	ANSI	Same as ANSI=T
	ANSI=0	No ANSI diagnostics
	omitted	Same as ANSI=0
	B=PUNCHB	Produce punched binary decks of all routines
	B=lfn	Put binary into a file
	B	Same as B=BIN
	B=0	No binary output
	omitted	Same as B=LGO
	BL	Burstable list (each major compilation section starts on a new page)
	BL=0	Compact list (new page for first page only
	omitted	Same as BL=0
	CS=FIXED	Collating sequence fixed weight table (display code)
	CS=USER	Weight table is user-defined by subroutines COLSEQ, WTSET, CSOWN
	CS	Same as CS=USER (at DTRC)
	omitted	Same as CS=FIXED (at DTRC)

```
DB=op/op/...  Debugging options
                (ARG=FIXED not allowed)
                op   meaning
                --   -------
                ER   generate code for object-time
                     reprieve of errors
                ID   generate output for interactive
                     debug (requires OPT=0)
                PMD  post mortem dump facility is
                     used
                SB   check subscript bounds
                SL   check character substring
                     expressions
                ST   same as DB=ID but no stylized
                     object code
                TB   full error traceback
DB          Same as DB=ER/ID/PMD/SB/SL/ST/TB
DB=0        All options deselected
            (DB=-ER/-ID/-PMD/-SB/-SL/-ST/-TB)
omitted     Same as DB=0  (if OPT=1,2,3)
            Same as DB=ER (if OPT=0)

DO=op/op    DO-loop interpretation
              op              meaning
             ----    --------------------------
             LONG    trip count may be > 131,071
             OT      at least once through each DO
                     loop
DO          Same as DO=OT
DO=0        Trip count must be <= 131,071 and
            minimum trip count is 0
omitted     Same as DO=0

DS          All C$ directives ignored
DS=0        All C$ directives processed
omitted     Same as DS=0

E=lfn       Output error list (see EL) on file lfn
E           Same as E=ERRS
omitted     Same as E=OUTPUT

EL=C        List catastrophic errors
EL=F        EL=C plus fatal errors
EL=W        EL=F plus warning errors
EL=T        EL=W plus trivial errors
EL          Same as EL=F
omitted     Same as EL=T
```

ET=C		Abort job if catastrophic errors during compilation; next control statement to be executed is the one after EXIT(S); if no EXIT(S), job ends
ET=F		Abort job if fatal or higher errors
ET=W		Abort job if warning or higher errors
ET=T		Abort job if trivial or higher errors
ET		Same as ET=F
ET=0		Continue even if compilation errors
omitted		Same as ET=F (at DTRC)
GO		Load and execute object code without a separate LGO (B=0 and QC not allowed)
GO=0		Do not load and execute
omitted		Same as GO=0
I=lfn		FORTRAN source input is in lfn
I		Same as I=COMPILE
omitted		Same as I=INPUT
L=lfn		Output lists (BL, LO) to file lfn (see also E option)
L=0		Listings are suppressed
L		Same as L=LIST
omitted		Same as L=OUTPUT
LO=op/op/...		Listing options (see L parameter)
	op	meaning
	--	-------
	A	list of variables, common blocks and attributes
	M	address map
	O	object code list (use only at request of User Services)
	R	cross-reference map
	S	source code list
LO		Same as LO=S/A/R
LO=0		No A, M, O, R, S information
omitted		Same as LO=S/A
MD=T		Flag machine-dependent usage (trivial)
MD=F		Flag machine-dependent usage (fatal)
MD=0		No machine-dependent diagnostics
omitted		Same as MD=0
OPT=0		Fast compile (required by DB=ID)
OPT=1		Intermediate optimization
OPT=2		High optimization, slow compile
OPT=3		OPT=2 plus potentially unsafe optimization
OPT		Same as OPT=2
omitted		Same as OPT=0

PD=8 Print density (E, L listings at 8 lines per inch)
PD=6 E, L listings at 6 lpi
PD Same as PD=8
omitted Same as PD=6

PL=<n> Print limit (decimal maximum number of records to be written at execution time on file OUTPUT (may be reset at execution time by specifying *PL=n, where n is the new line limit, at the end of the execute statement (max: 9 999 999 999)
PL Same as PL=50000
omitted Same as PL=5000

PN Page numbering of output list is continuous
PN=0 Each subprogram starts with page 1
omitted Same as PN=0

PS=n Page size (number of lines per page in compilation listing) (n >= 4)
omitted Same as PS=80 (if PD=8)
Same as PS=60 (if PD=6)

PW=n Page width (number of characters per line in compilation listing) (50 <= n <= 136)
PW Same as PW=72
omitted Same as PW=72 (if L or E file is connected)
Same as PW=136 (all other files)

QC Quick syntax check (no binary output or cross reference addresses) (conflicts with B,GO,LO=O/M)
QC=0 No quick syntax check
omitted Same as QC=0

REW=op/op/... Rewind option

op	meaning
B	rewind binary output file (object code)
E	rewind error file
I	rewind input file
L	rewind output file

REW Same as REW=I/B
REW=0 No files rewound
omitted Same as REW=0

```
                   ROUND=op/op/...  Rounded arithmetic for
                            specified operators
                            (op is one of: A, S, M, D, for +, -, *,
                            and /, respectively)
                   ROUND    Same as ROUND=A/S/M/D (at DTRC)
                   ROUND=0  Rounded arithmetic not used
                   omitted  Same as ROUND=A/S/M/D (at DTRC)

                   SEQ      Sequenced line format
                   SEQ=0    Standard FORTRAN format
                   omitted  Same as SEQ=0

        Defaults:  FTN5,ANSI=0,B=LGO,BL=0,CS=FIXED,DB=ER,DO=0,DS=0,
                       E=OUTPUT,EL=T,GO=0,I=INPUT,L=OUTPUT,LO=S/A,
                       OPT=0,PD=6,PL=5000,PN=0,PS=60,PW=136,QC=0,
                       REW=0,ROUND=A/S/M/D,SEQ=0.

        Similar commands:  Cray:     CFT; CFT77
                           NOS/BE:   FTN5; FTN4
                           VMS:      FORTRAN

        Examples:  FTN5,I=myprog,L=0,GO.    <-- compile and go with
                                                no listing


GET     Get copies of indirect permanent files as local files.

        Syntax:    GET,lfn_1=pfn_1,lfn_2=pfn_2,...,lfn_n=pfn_n
                      /UN=un,PW=pw,NA,RT.

        See also:  ATTACH, FETCH (Cray)

        Examples:  GET,myindfl.
                   GET,herindf/UN=her_un,PW=her_pw.


GO      Clear the pause bit of one of your jobs.

        Syntax:    GO,jsn.

        Remarks:   The pause bit may be set by one of your programs
                   or by the PAUSE command.

        See also:  PAUSE

        Similar commands:  NOS/BE:  GO

        Examples:  GO,AAXJ.
```

GOODBYE Terminate an application.

Syntax: `GOODBYE,application`

Remarks: GOODBYE can appear in a procedure or a batch job, where it terminates the job.

See also: BYE, LOGOUT; HELLO, LOGIN

Similar commands:

Cray:	^Z, QUIT
NOS/BE, VMS:	LOGOUT

Examples: `GOODBYE`

GTR Selective extraction of records from a (library) file.

Syntax: `GTR,lfn_1,lfn_2,d,NR,S,NA.dir_1,dir_2,...,dir_n`

Parameters:

lfn_1 - source file
(Default: OLD)

lfn_2 - output file (must be disk if "d" used)
(Default: LGO, positioned at EOI (disk)
or BOI (tape, unless NR)

d - random access directory

- omitted - no new random access directory; if ULIB type, first record not copied, rest of records copied; last record of OPLD copied without alteration; no EOF written at end of file
- U - no new random access directory; if ULIB type, first record copied without alteration with rest of library and OLPD; no EOF written at end of file
- D - write random access directory on lfn_2 with entries for selected records; if ULIB type, first record copied without alteration with rest of library and OPLD; EOF written after new directory
- other - same as D

NR - do not rewind lfn_1 after; do not rewind lfn_2 before or after; copy an existing directory from lfn_1 to lfn_2

S - search lfn_1 sequentially

NA - do not abort on error

dir_i - a record or group of records to get:

```
type/name
name
type_1/name_1-type_2/name_2
type_1/name_1-name_2
name_1-name_2
type/name-* (all type from name on)
name-*      (all from name on)
type/*      (all of type)
*           (all records)
0           (insert zero-length record)
```

See also: LIBEDIT

```
Similar commands: Cray:    BUILD
                  NOS/BE:  COPYN (sequential only)
                  VMS:     LIBRARIAN
```

```
Examples:    GTR,SYSTEM,BIN,D.PP/*
                          ^-- copy all PP records, build random
                              access directory
             GTR,OPL,NEW,,NR.OPLC/COMCARG,0,COMCCIO
                          ^-- copy 2 records and put a zero-
                              length record in between them at
                              the current position of NEW; NEW
                              not rewound, OLD rewound before
             GTR,SYSTEM,SYSLIB,D.ULIB/SYSLIB
                          ^-- user library SYSLIB copied from
                              file SYSTEM to end of SYSLIB
             GTR.REL/A  <-- copy record A from OLD to LGO
```

HELLO (IAF) Logs you out of IAF and switches you to another application, or starts another login.

Syntax: HELLO,application

Remarks: If IAF is a secondary application, HELLO,appl is the same as BYE,appl.

If application is omitted, a new login is started.

```
         See also:    LOGIN; BYE, GOODBYE, LOGOUT

         Similar commands:  NOS/BE:  LOGIN
                            VMS:     LOGOUT then redial

         Examples:    HELLO,ICF     <-- switch to Interactive Cray
                                        Facility


HELP     (IAF) Ask for help.

         Syntax:      HELP.

         Remarks:     Displays a menu and prompts for your selection.

                      Help features:
                         . list of all NOS commands, except compiler
                           calls
                         . help in entering a command
                         . access to on-line CDC manuals
                         . list of NAM/CCP network commands

         See also:    EXPLAIN, HELPBE, HELPME

         Similar commands:  NOS/BE:  BEGIN,DOCGET (DTRC)
                            VMS:     HELP

         Examples:    HELP.


HELPBE   On-line help for the NOS-equivalent of NOS/BE commands.

         Syntax:      HELPBE.

         Remarks:     HELPBE is in library BETONOS/UN=LIBRARY.

         Examples:    ATTACH,USRLIB8=BETONOS/UN=LIBRARY.
                      LIBRARY,USRLIB8/A.
                      HELPBE.            <-- you will be prompted
                      LIBRARY,USRLIB8/D.


HELPME   (IAF) Display a brief description of a command, prompt for
         parameters, execute the command.

         Syntax:      HELPME,command

         Parameters:  command - the command you want help with
```

Remarks: This is an interactive procedure. Enter a question mark (or the HELP function key in screen mode) at any time for help during the dialog.

See also: EXPLAIN, HELP, HELPBE

Similar commands:

NOS/BE:	Interactive procedures
VMS:	HELP @COS (just the description)
VMS Cray Station:	HELP @COS (just the description)

Examples: HELPME,FCOPY.

IF Conditionally skip one of more commands.

Syntax:

```
IF,condition.command.   <-- note two terminators

IF,condition,label.
```

Synonym: IFE

Parameters:

condition - an expression evaluating to true or false

command - a valid command

label - alphanumeric string (1-10 characters, starting with a letter)

Similar commands:

Cray, VMS:	IF
NOS/BE:	IFE

Examples:

```
IF(R1=1,there)
...
ENDIF(there)
= = = = =
IF,R1=1.REWIND,fyle.
```

ITEMIZE List information about each record of a binary file.

Syntax: ITEMIZE,lfn,L=listfyl,BL,PW=n,PD,NR,N=n,E,U.

Parameters: params - optional parameters:

BL - burstable listing (new page for each file output)

E - list entry points for relocatable modules; list IDENTs for UPDATE sequential PL

```
L=     - the output listing, if other
         than OUTPUT
N      - process to end-of-information
N=n    - number of files to process
         (default: 1)
NR     - lfn not rewound before or
         after
PD     - print density is 8 lpi
         (default: 6 lpi)
PW=pw  - page width
         (defaults:
          batch: 136 or 72
          IAF:    72 (output file
                      connected)
                 136 (output file not
                      connected)
U      - itemize all records in a user
         library
         (default: only user library
                   directory)
```

Remarks: Output includes record number, name, length, prefix table for relocatable binary or user library.

For a sequential UPDATE PL, only deck names are listed.

Your terminal should be in NORMAL mode (not ASCII) before listing ITEMIZE output at your terminal.

Defaults: ITEMIZE,LGO,L=OUTPUT,N=1,PW=see above.

See also: CATALOG

Similar commands: Cray: ITEMIZE
NOS/BE: ITEMIZE; LISTBIN (DTRC)
VMS: LIBRARIAN

Similar commands:	Cray:	ITEMIZE
	NOS/BE:	ITEMIZE; LISTBIN (DTRC)
	VMS:	LIBRARIAN

Examples:

```
ITEMIZE,oldpl.
ITEMIZE,userlib,U,L=uout.
```

job Identifies requirements for a batch job.

Syntax:

```
ujn,SC=sc,T=t,CM=cm,ST=lid.punchmode
ujn,SCsc,Tt,CMcm,STlid.punchmode
ujn,sc,t,cm,lid.punchmode

ujn,P=p,T=t,CM=cm,ST=lid.punchmode
ujn,Pp,Tt,CMcm,STlid.punchmode
ujn,p,t,cm,lid.punchmode
```

Parameters: sc - do not use at DTRC

p - do not use at DTRC

t - job step time limit in CPU seconds
(Range: 1 to 32767 decimal;
1 to 77777B octal)
(Default: 64 decimal)

cm - maximum octal field length required
(Maximum: 376500 octal)
(Default: 376500 octal)

lid - not used at DTRC

punchmode - in columns 79-80 of actual punched cards
26 - 026 mode
29 - 029 mode

Similar commands: Cray: JOB
NOS/BE: job statement

Examples:

```
ABCDjbl,CM75000,T10.        <-- these three
ABCDjbl,CM=75000,T=10.      <--  are the
ABCDjbl,,10,75000.          <--   same
- - - - -
ABCD.                       <-- all defaults
```

LABEL Mount a magnetic tape and, if labelled, check the label.

Syntax:

```
LABEL,lfn,VSN=vsn_1/vsn_2=...=vsn_n-1/vsn_n,
      MT¦NT,D=den,F=format,LB=lb,
      FC=fcount,CV=cv,NS=ns,PO=plp2...pn,CK¦CB,
      SI=setid¦M=setid,SN=secno,QN=seqno,
      L=fileid,FA=fa,G=genno,E=gvn,
      CR=cdate¦C=cdate,RT=yyddd¦T=ddd,W¦R,
      AC=ac,CT=ct,MD=md,PW=pw,
      TO=to,UN=username.
```

Required parameter: lfn - local file name for the tape - if lfn is already a local disk file, processing continues - if lfn is already a mounted tape and R is present, the label is checked

Parameters: AC= - alternate auditability

CK - lfn is to be used as a checkpoint file
CB
CK - append dump to lfn
CB - write dump at BOI of lfn

CR= - creation date (yyddd)
C=

CT= - file category

CV=cv - conversion mode for 9-track labels (do not use with MT)

cv	meaning
AS	ASCII/6-bit display code
US	same as AS
EB	EBCDIC/6-bit display code

D=den - tape density

MT		NT	
den	density	den	density
LO	200 cpi	HD	800 cpi
HI	556 cpi	PE	1600 cpi
HY	800 cpi	GE	6250 cpi
200	200 cpi	800	800 cpi
556	556 cpi	1600	1600 cpi
800	800 cpi	6250	6250 cpi

E= - 1- to 4-digit generation version number (Default: 0)

F= - data format

format	meaning
I	internal
SI	system internal (NOS/BE tape)
S	stranger
L	long block stranger
F	foreign (unknown data format)

(Default: F=I)

FA= - File accessibility character

fa	meaning
blank	unlimited access
A	only the owner can access it
other	future accesses must specify this character

FC= - maximum block size in frames (required if F=F is specified)

G= - 1- to 4-digit generation number (Default: 1; 0 not allowed)

```
L=       - 1- to -17-character file label
           (Default: blank)

LB=lb      labelled or unlabelled tape
             lb   meaning
             --   -------
             KL   ANSI-labelled
             KU   unlabelled
             NS   nonstandard-labelled
                  (assumes data starts immediately
                  after the first tape mark)

MD=      - file mode

MT       - 7-track tape
NT       - 9-track tape

NS=ns    - noise size (any block shorted than ns
           frames is discarded); not valid for F=I,
           F=SI; D=PE, or D=GE
           (Maximum: 31; NS=0 ==> the default;
            Defaults: 0 (PE,GE), 18 (others))

PO=      - processing options
             po   meaning
             --   -------
             R    Read the tape (ring OUT)
             W    Write the tape (ring IN)
           Several other options are available.

PW=      - password

QN=      - 1- to 4-digit file section number of the
           file within a multivolume file set
           (Default: 1; use QN=9999 to append a new
           file to a multifile set)

R        - Read the label and check specified
           fields
W        - write a new label, if the existing label
           has expired
           (Default: R; ignored for F=SI and QN>1)

RT=      - expiration date (yyddd)
T=       - retention number of days (0 to 999)

SI=      - 1- to 6-character file set identifier
           (required for file positioning in multi-
           file set)
```

```
                  SN=     - 1- to 4-digit file section number of the
                            volume within a multivolume file set
                            (Default: 1)

                  TO=     - TMS option

                  UN=     - user name

                  VSN=    - 1- to 6-character volume serial number
                              / - separates multiple reels
                              = - use first available VSN
                            (Default: from separate VSN statement
                                      preceding LABEL)

         See also:        VSN

         Similar commands:  Cray:     ACCESS
                            NOS/BE:   LABEL
                            VMS:      REQUEST,MOUNT

         Examples:        LABEL,tbe,VSN=NA9999,PO=R,F=SI,LB=KU,D=GE.
                                         ^-- read a NOS/BE unlabelled tape
                                             at 6250 cpi


LDSET    (Loader)  Set option(s) for the current load.

         Syntax:          LDSET(opt1,opt2,...,optn)
                          LDSET.

         Parameters:      opti - option in one of the following forms:
                                   key
                                   key=param
                                   key=param1/param2/.../paramn
                                 options include:
                                   FILES,lfn1/lfn2/.../lfnn
                                     probably not needed

                                   LIB=lib1/lib2/.../libn
                                     LIBEDIT libraries to be searched

                                   MAP=p/lfn
                                     p    - one or more of:
                                                N - no map
                                                S - error messages and loader
                                                    statistics
                                                B - block list and list of
                                                    unsatisfied externals
                                                E - entry point list without
                                                    cross reference
```

X - entry point list with cross reference
(default: SB)

lfn - the file to hold the map
(default: OUTPUT)

PRESET=p
PRESETA=p
preset memory as specified:

p	octal preset value
NONE	no presetting
ZERO	0000 0000 0000 0000 0000
DEBUG	6000 0000 0004 0040 0000
NGINF	4000 0000 0000 0000 0000
ALTZERO	2525 2525 2525 2525 2525 (alternating 0 and 1)
n	a 1-20 digit octal constant with optional +, - and terminal B

(PRESETA puts each location's address in the 17 low order bits)
(default: PRESET=ZERO)

Remarks: MAP=S provides *statistics*: program length, routines present and missing.

LDSET without parameters will prompt you for them.

See also: page 5-6-5

Similar commands:		
	Cray:	SEGLDR,PRESET=
	NOS/BE:	LDSET, XEQ
	VMS:	virtual memory

Examples:

```
LDSET,PRESETA=DEBUG,MAP=S.
              ^-- the NOS/BE PRESET default
= = = = =
/LDSET
LDR>? LDSET,MAP=S,PRESETA=DEBUG.
LDR>? LDSET,LIB=DTLIB.     <-- previously attached
LDR>? LOAD,LGO.
LDR>? NOGO,ABS.            <-- create absolute
/ABS.                      <-- execute
```

LENGTH Gives the current status of one of your local files.

Syntax: LENGTH,lfn.

Remarks: The information includes is length (PRUS), type, current status.

See also: ENQUIRE,FN=lfn.

Similar commands: Cray: DS
NOS/BE: FILES

Examples: LENGTH,mylfn.

LGO (Loader) Load and execute the default compiler binary output file.

Syntax: LGO.
LGO,plist.

Parameters: plist - list of positional and/or keyword parameters for the program being executed

See also: name

Similar commands: Cray: name
NOS/BE: LGO

Examples: FTN5.
LGO.

LIBEDIT Create and maintain a library of programs, subprograms, procedures, or text.

Syntax: LIBEDIT,P=lfn,N=lfn,I=lfn,Z,B=lfn,L=lfn,
LO=options,V,C,D,U=record,NA,NI,NR,NX=n.

Parameters: P= - the old file
P=0 => create new file from replacement files

N= - the new file

I= - input directive file
I=0 => no directives

Z - directives immediately follow the command terminator (the first character after the terminator is the directive separator; overrides I=)

B= - replacement and insertion records

B=0 => no replacement file

L - summary of LIBEDIT run and any requested listings

L=0 => suppress output

LO= - list options

option	meaning
C	list directives
E	list errrors
M	list modifications
N	list records written to new file
F	full listing (same as LO=CEMN)

V - verify new file against old file by calling VFYLIB (U overrides V)

C - copy new file over old file when done

D - do not abort on processing errors

U= - old file is a user library; new file is made a user library by call to LIBGEN (overrides V) - the value <record> becomes the name of the user library directory record

U or omitted - same a ULIB is used

NA - do not abort on directive errors

NI - do not insert unreplaceable records at EOF of new file

NR - old and new files not rewound before or after

NX= - n=0 - include cross references in library directory of new user library

n<>0 - do not include cross references (has meaning only for U or the *LIBGEN directive)

Directives: See Section 5-5.

Defaults:

```
interactive: LIBEDIT,P=OLD,N=NEW,I=INPUT,B=LGO,
                         L=OUTPUT,LO=EM,NX=0.
all others:  LIBEDIT,P=OLD,N=NEW,I=INPUT,B=LGO,
                         L=OUTPUT,LO=F,NX=0.
```

Remarks: If the output it to tape and the tape may be processed at a later date by GTR or MODIFY, put the new file on disk and COPY it to tape. This will insure that the directories have disk PRU random addresses and not tape PRU random addresses.

See also: LIBGEN

Similar commands:

Cray:	BUILD
NOS/BE:	EDITLIB, COPYN
VMS:	LIBRARIAN

Examples: LIBEDIT,F=LGO,P=mysubs.

LIBGEN Create a new user library of routines for use by the loader.

Syntax: LIBGEN,F=lfn,P=lfn,N=name,NX=n.

Parameters:

F= - source file containing absolute (ABS), overlay (OVL), procedure (PROC), relocatable (REL), or capsule (CAP) records (no library generated if none of these records is in F)

P= - will contain the new user library

N= - name of the generated user library; entered in ULIB and OPLD records (default: P=lfn)

NX= - n=0 - include cross references in library directory of new user library
n<>0 - do not include cross references
(default: NX=0)

See also: LIBEDIT

Similar commands:

Cray:	BUILD
NOS/BE:	EDITLIB
VMS:	LIBRARIAN

Examples:

```
ATTACH,subs.
FTN5,I=subs.
LIBGEN,P=myabs.
```

LIBLOAD (Loader) Load modules from a library which contain the specified entry points.

Syntax: LIBLOAD,libname,etpname1,eptname2,...,eptnamen.

Parameters: libname - the library containing the desired entry points

eptnamei - a specific entry point to be loaded

Remarks: For a core image load, only one entry point may be given.

See also: LOAD

Similar commands: NOS/BE: LIBLOAD
VMS: LINK ...,library/LIB/INCLUDE=

Examples: LIBGEN,F=LGO,P=mysubs.

LIBRARY (Loader) Specify a set of global libraries to be searched for externals and programs and the order in which they are to be considered.

Syntax: LIBRARY,file1,file2,...,filen/directive.

Parameters: filei - system or user library (a maximum of 2 user libraries)

directive - specify if the files are to be added to, deleted from, or replace your global library set.

directive	meaning
A	add
D	delete
R	replace

(default: R)

Omit all parameters to clear your global library set.

Remarks: The order of search for externals is:
global (those on most recent LIBRARY)
local (those in LDSET,LIB= or in LDSET tables in the loaded modules)
system (SYSLIB)

The order of search for programs is:
  local files; global, local and system (NUCLEUS)
  libraries

LIBRARY may not occur in a load sequence.

A no-auto-drop status is set for these files while they are in the global set. See SETFS.

```
Similar commands:  Cray, NOS/BE:  LIBRARY
                   VMS:           LINK ...,library/LIB

Examples:      LIBRARY,MYLIB.          <-- global set has MYLIB

               LIBRARY,YOURLIB/A.      <-- global set has MYLIB
                                            and YOURLIB
               LIBRARY.                <-- global set empty
```

LIMITS List your validation limits.

Syntax: LIMITS,L=lfn.

Similar commands: NOS/BE: ASSETS

Examples: LIMITS.

LINE (IAF) Set your terminal for line mode.

Syntax: LINE.

LINE,TM=model.

LINE,model.

Parameters: See SCREEN.

Remarks: LINE may be included in a procedure.

LINE is the default setting unless SCREEN is included in your LOGINPR file.

Affects FSE, HELPME, screen formatting, and the display of NOS procedure parameteis.

See also: SCREEN

Examples: LINE.

```
LIST     (IAF) List lines of a local file.

         Syntax:      LIST,F=lfn

         Parameters:  F=     - the local file to be listed
                                (default: the primary file)

         Similar commands:  NOS/BE:  COPYSBF, COPYSF, COPYSR, LISTN,; TYPE·
                                     LISTZ (last 4 DTRC)
                            VMS:     VSYS:LISTN (DTRC)

         Examples:    LIST,F=MYFILE
                                 ^-- list local file MYFILE


LISTLB   List labels of an ANSI-labelled tape.

         Syntax:      LISTLB,lfn,SI=setid,QN=seqno,LO=ltype,L=out.

         Parameters:  SI= - 1- to 6-character file set identifier

                      QN= - 1- to 4-character file sequence identifier

                      LO= - label type(s) to be listed
                              A - all labels
                              R - required labels
                              O - optional labels
                              V - VOLn labels
                              H - HDRn labels
                              F - EOFn labels
                              E - EOVn labels
                              U - uvln, uhln, utln labels
                            (default: A)

         Similar commands:  NOS/BE:  LISTMF

         Examples:    LABEL,tape,....
                      LISTLB,tape.
```

```
LISTLID  List network configuration and host availability information.

         Syntax:     LISTLID,LID=lid,PID=pid,L=lfn.
                     LISTLID,ST=lid,PID=pid,L=lfn.

         Parameters: LID= - a specific logical identifier
                     ST=

                     PID= - a specific physical identifier

         Similar commands: NOS/BE:  Q,ID
                           VMS:     SHOW NETWORK

         Examples:   LISTLID.


LOAD     (Loader) List of object files whose contents are to be loaded.

         Syntax:     LOAD,lfn1,lfn2,...,lfnn.

         Parameters: lfni    - rewind (except INPUT) before loading
                     lfni/R  - rewind before loading
                     lfni/NR - do not rewind before loading

         See also:   LIBLOAD, SLOAD

         Similar commands: Cray:    SEGLDR BIN=dn1,dn2,...
                           NOS/BE:  LOAD
                           VMS:     LINK f1,f2,...

         Examples:   LOAD,LGO,BIN.


LOCK     Prevent writing on a file.

         Syntax:     LOCK,lfn1,lfn2,...,lfnn.

         Parameters: lfni - a local file

         Remarks:    Used to prevent writing on a local file.

         See also:   UNLOCK

         Similar commands: Cray:
                           NOS/BE:  ATTACH,...,MR=1
                           VMS:     OPEN(...,READONLY) in Fortran
                                    program
```

```
Examples:      ...             <-- create a new file
               LOCK,newfile.
                               ^-- inhibit further writing on file
                                   NEWFILE
               ...             <-- other commands
               UNLOCK,newfile.
                               ^-- all writing on file NEWFILE
```

LOGIN (IAF) Terminate your current application and start another.

Syntax: LOGIN,application

Remarks: LOGIN may be used in a procedure or batch job, where it terminates the job.

See also: HELLO; LOGOUT, BYE, GOODBYE

```
Similar commands:  NOS/BE:  LOGIN (not in a procedure or
                                   batch job)
```

```
Examples:      LOGIN,ICF       <-- switch to ICF
```

LOGOUT (IAF) Terminate an application.

Syntax: LOGOUT,application

Remarks: LOGOUT may be used in a procedure or batch job, where it terminates the job.

See also: BYE, GOODBYE; LOGIN, HELLO

```
Similar commands:  Cray:    ^Z,QUIT
                   NOS/BE:  LOGOUT (not in a procedure or
                                    batch job)
                   VMS:     LOGOUT
```

Examples: LOGOUT

LO72 Reformat files.

Syntax: LO72,p1,p2,...,pn.

```
Parameters:    I=lfn    - file with reformat parameters
               I        - same as I=INPUT
               I=0      - no file of reformat parameters
               omitted  - same as I=0
```

```
          S=lfn   - input file to be reformatted
          S       - same as S=SCR
          omitted - same as S=SCR

          L=lfn   - output reformatted file
          L       - same as L=OUTPUT
          omitted - same as L=OUTPUT

          T=x     - type of file being reformatted
                      x        meaning
                      -   -----------------
                      M   Modify source data
                      C   COMPASS source data
                      B   other source data
          T       - same as T=B
          omitted - same as T=B

          H=xxx   - (truncation) length of output line
          H       - same as H=72
          omitted - same as H=72
                    (max: 150; must be >= Nx+Ox)
                    (see Remarks below)

          LP      - format for line printer

          NR      - do not rewind S file

          Nx=y    - number of characters to be moved
                     x - field number (1-6)
                     y - number of characters being moved
                    (see Remarks below)

          Ix=y    - input data field
                     x - field number (1-6)
                     y - starting column
                    (see Remarks below)

          Ox=y    - output data field
                     x - field number (1-6)
                     y - starting column
                    (see Remarks below)

          IT      - suppress query before each change
          omitted - query before each change
                    (interactive jobs only)

Remarks:  Restrictions on H, N, I, O:
            (Nx+Ix)>150 --> error for 1<=x<=6
            (Nx+Ox)>H   --> error for 1<=x<=6
            H      >150 --> error
```

Defaults for N, O, I:

type	N1	I1	O1	N2	I2	O2	N3	I3	O3
B	72	1	1	0	0	0	0	0	0
C	7	9	1	50	41	8	15	112	58
M	2	6	1	48	10	3	22	82	51

Ni,Ii,Oi=0 for 4<=i<=6.

Most useful in compressing compiler list output to fit into 72 columns

Similar commands: NOS/BE: COPYEXT; COPYS (both DTRC)
VMS: VSYS:CPYEXT (DTRC)

Examples:

```
LO72,S=myin,L=myout,I1=2,O1=1.
              ^-- restore a file that was shifted
                  one column to the right, perhaps
                  by COPYSBF
```

MAP (Loader) Specify the global default option for load maps.

Syntax:

```
MAP.
MAP,p.
```

Parameters: p - the desired load map

p	meaning
OFF	no map (same as LDSET,MAP=N)
PART	statistics and block map (same as LDSET,MAP=SB)
ON	PART plus entry point cross-reference (same as LDSET,MAP=SBX)
FULL	ON plus entry point map (same as LDSET,MAP=SBEX)

(default at DTRC: OFF)

Remarks: MAP without a parameter resets to the default.

MAP remains in effect until changed by another MAP statement. It may be overridden for the next load by using LDSET,MAP=.

The more map requested, the more CP time and memory is required to generate it.

See also: LDSET

```
         Similar commands:  Cray:     SEGLDR
                            NOS/BE:   MAP
                            VMS:      LINK qualifiers

         Examples:      MAP(PART)


MERGE    Merge files.

         Syntax:        FILE,lfninl,....
                        FILE,lfnin2,....
                          ...
                        FILE,lfnout,....
                        MERGE.pl,p2,...,pn
                           or
                        MERGE.pl p2 ... pn

                        Positional:      MERGE.from,to,key,dir,l,,e,el,
                                               dialog,end,,,ownf,ownfl,
                                               ownmrl,,ownl,own2,own3,
                                               own4,own5,retain,seqa,seqn,
                                               seqr,seqs,status,sum,,
                                               verify,fastio.

                        Interactive:     MERGE.DIALOG=YES

                        Directive file:  MERGE.DIR=lfn
                                         MERGE.params,DIR=lfn
                                         MERGE.DIR=lfn,params
                                         MERGE.params,DIR=lfn,..
                                              .more_params

         Parameters:    See SORT5.

                        VERIFY - Verify that each input file is sorted
                                 before merging them.

         Remarks:       Files to be merged must be presorted.

                        See SORT5 remarks.

         See also:      SORT5

         Similar commands:  Cray, VMS:  SORT
                            NOS/BE:     MERGE

         Examples:      FILE,inl,BT=C,RT=Z,MRL=80.
                        FILE,in2,BT=C,RT=Z,MRL=80.
                        FILE,outfyl,BT=C,RT=Z,MRL=640.
                        MERGE.(inl,in2),outfyl
                                      ^-- merge two files
                        MERGE.FROM=(inl,in2),TO=outfyl
                                      ^-- same
```

MFL Reset maximum field length for subsequent job steps.

Syntax: MFL,CM=nnnnnn.
MFL,nnnnnn.

Remarks: MFL clears RFL and allows the system to determine the FL for each job step.

MFL cannot exceed the job statement CM or 376500 octal, whichever is lower.

See also: RFL.

Similar commands: NOS/BE: EFL; RFL

Examples: MFL,200000.

MODE Mode error bypass should not be used at DTRC. An attempt to ignore Error Mode 1 may cause an Error Mode 0.

MODIFY Edit a Modify-formatted program library.

Remarks: Use UPDATE.

name (Loader) Load and execute binary program or procedure in local file "name".

Syntax: name.
name,plist.

Parameters: plist - list of positional and/or keyword parameters for the program or procedure being executed

See also: BEGIN, LGO

Similar commands: Cray: name
NOS/BE: name
VMS: RUN

Examples: ATTACH,myprog.
myprog.

NOEXIT Continue processing with the next command even if an error has occurred (suppress EXIT processing).

Syntax: NOEXIT.

See also: EXIT, ONEXIT

Similar commands: NOS/BE: EXIT,U
VMS: ON condition

Examples:

```
NOEXIT.      <-- Exit processing off
FTN5.
ONEXIT.      <-- restore exit processing
LGO.         <-- executed even if compile errors
...          <-- not executed if execution errors
```

NOGO (Loader) Complete loading of a program, generate load map (if requested), put absolute into a file (if requested), but do not execute.

Syntax: NOGO.
NOGO,abs.

Parameters: abs - will contain the loaded program as a single core image module (non-segmented/non-overlay loads only)
(<abs> is suitable for inclusion in a LIBEDIT library)

See also: LDSET

Similar commands: Cray: SEGLDR (ABS= directive)
NOS/BE: NOGO
VMS: LINK

Examples:

```
DEFINE,myprog/NA.
LOAD,LGO.
NOGO,myprog.
```

NORERUN Clear the job rerun status.

Syntax: NORERUN.

Remarks: May be useful to prevent updating a file when the job would ordinarily be rerun.

See also: RERUN

Similar commands: Cray: RERUN
NOS/BE: NORERUN

Examples: `NORERUN.`

NORMAL (IAF) Reverse the effect of ASCII, AUTO, BRIEF, CSET,ASCII, and NOSORT commands.

Syntax: `NORMAL`

See also: ASCII, AUTO,BRIEF, CSET, NOSORT

Examples: `NORMAL`

NOTE Create a file with the command line containing the lines for the new file.

Syntax: `NOTE,lfn,NR./line_1/line_2/.../line_n`

Parameters:

lfn	- (default: OUTPUT)
NR	- Do not rewind lfn before and after (default: rewind)
/	- a delimiter (any character) denoting the start of a new line for the file (the character immediately following the terminator is the delimiter)
line_i	- the contents of the i-th line of the new file

Remarks: The NOS/BE default is NO rewind.

Similar commands:

Cray:	NOTE
NOS/BE:	NOTE (DTRC)
VMS:	OPEN,WRITE,CLOSE

Examples:

```
NOTE,DATA./  1  2.4/LINE OF TEXT/0.1 1E-4/END
                  ^-- create a new file DATA
Local file DATA contains:
  1  2.4
LINE OF TEXT
0.1 1E-4
END
- - - - -
```

```
NOTE,DATA,NR./  1  2.4/LINE OF TEXT/0.1 1E-4/END
NOTE,DATA,NR./  2  3.6/ANOTHER LINE OF TEXT
NOTE,DATA,NR./0.1 1E-4/END
                 ^-- create file with many lines
PACK,DATA.       <-- remove embedded EORs
Local file DATA contains:
  1  2.4
LINE OF TEXT
0.1 1E-4
END
  2  3.6
ANOTHER LINE OF TEXT
0.1 1E-4
END
= = = = =
NOTE,UIN./*compile progl,subl,sub2
UPDATE,I=UIN.
= = = = =
NOTE./THE PROGRAM FINISHED
             ^-- useful for displaying messages
                 (comments) from a procedure
```

NULL (IAF) Select the NULL subsystem.

Syntax: NULL

Remarks: This is the default subsystem in a batch job.

RUN will not work in the NULL subsystem.

Examples: NULL

OFFSW Clear sense switches.

Syntax: OFFSW,switch_1,switch_2,...,switch_n,jsn.

Parameters: switch_i - a sense switch to be cleared (1-6)
0 - clear all sense switches

jsn - since a jsn is recognized by its alphabetic characters, jsn may appear anywhere in the parameters list

See also: ONSW

Similar commands: Cray, NOS/BE: SWITCH

Examples:

```
OFFSW,0,ABCD.
             ^-- clear all sense switches for job
                 ABCD
```

ONEXIT Reverse the effect of NOEXIT.

Syntax: ONEXIT.

See also: EXIT, NOEXIT

Similar commands: VMS: NOON; ON condition THEN CONTINUE

Examples: See NOEXIT.

ONSW Set sense switches.

Syntax: ONSW,switch_1,switch_2,...,switch_n,jsn.

```
Parameters: switch_i - a sense switch to be set (1-6)
                         0 - set all sense switches

            jsn      - since a jsn is recognized by its
                       alphabetic characters, jsn may appear
                       anywhere in the parameters list
```

See also: OFFSW, SWITCH

Similar commands: Cray, NOS/BE: SWITCH

```
Examples:   ONSW,ABCD,4,5.
                         ^-- turn sense switches 4 and 5 on
                             in job ABCD
```

OPLEDIT Remove modification decks and identifiers from a MODIFY library.

Remarks: Use UPDATE instead of MODIFY.

OUT Send deferred output files to the print or punch queue immediately.

```
Syntax:     OUT.                          <-- queue all files

            OUT,*,lfn1,lfn2,...,lfnn. <-- queue all files,
                                          except those listed
```

Parameters:

Remarks: OUT processes any file given deferred ROUTE-ing as well as OUTPUT, PUNCH, PUNCHB, P8.

See also: ROUTE

```
        Similar commands:  Cray:    DISPOSE
                           NOS/BE:  ROUTE

        Examples:    OUT.


OVWRITE Overwrite files to erase (destroy) their contents.

        Syntax:      OVWRITE,lfn1,lfn2,...,lfnn/OP=p1p2.
                                  ^-- overwrite specified files

                     OVWRITE,*,lfn1,lfn2,...,lfnn/OP=p1p2.
                                  ^-- overwrite all but specified
                                      files

        Parameters:  OP= - how files are to be overwritten and
                           whether they are to be released
                            pi              meaning
                            --    ------------------------------
                            Z     overwrite with zeros
                            X     overwrite with zeros, then ones,
                                  then alternating zeros and ones
                            R     release files after overwriting
                           (default: OP=Z)

        Similar commands:  Cray:    SCRUBDS; WRITEDS

        Examples:    OVWRITE,fyl1,OP=XR.
                                  ^-- clear a file, then release it


PACK    Combine all records/files in a file by removing all EORs and
        EOFs.

        Syntax:      PACK,lfn_in,lfn_out,x.

        Parameters:  lfn_in  - (not rewound after)

                     lfn_out - (default: lfnout=lfnin;
                               rewound after, but not before)

                     x       - non-null to not rewind lfn_in before
                               packing

        Remarks:     Do not use with S, L, or F tapes

        Similar commands:  NOS/BE:  COMBINE

        Examples:    PACK,infyl,pkdfyl.
```

```
PASSWOR Change your password.

        Syntax:     PASSWOR,oldpw,newpw.
                    PASSWOR.

        Parameters: oldpw - old password
                    newpw - new password (4-7 characters)

        Remarks:    Must be set separately for batch and interactive.


        Similar commands: Cray:    ACCOUNT,NUPW=nupw,...;
                                   NEWCRAYPW (from VMS)
                          NOS/BE:  interactive: TURNKEY
                                   batch:       none
                          VMS:     SET PASSWORD

        Examples:   PASSWOR,old,new.


PAUSE   Set the pause bit of one of your executing jobs.

        Syntax:     PAUSE,jsn.

        See also:   GO

        Examples:   PAUSE,ABCD.


PERMIT  Explicitly permit another user to access one of your private
        or semi-private files.

        Syntax:     PERMIT,pfn,un_1=m_1,un_2=m_2,...,un_n=m_n/NA.

        Similar commands: Cray:  permit lists
                          VMS:   Access Control Lists

        Examples:   PERMIT,myfile,ABCD=R.
                                  ^-- allow ABCD to read the file


PURGALL Purge all your files which match the parameters.

        Syntax:     PURGALL,TY=ty,CT=ct,AD=ad,MD=md,CD=cd,AF,TM=tm,NA.

        Parameters: ty - file type
                          ty              meaning
                       ----------   --------------------------
                       I (INDIR)    all indirect files
                       D (DIRECT)   all direct files
                       A (ALL)      all files
                      (Default: TY=A, if any other parameter is
                                specified)
```

ad - all files last accessed before (after, if AF) this date (yymmdd)

md - all files last modified before (after, if AF) this date (yymmdd)

cd - all files created before (after, if AF) this date (yymmdd)

AF - all files after AD, MD, or CD dates

tm - time-of-day on the AD, MD, CD date (hhmmss)

Remarks: AF, CT, DN, MA, TY, TM, and one date (either AC, MD, CD) fit on a single PURGALL command.

Similar commands: VMS: DELETE; PURGE

Examples:

```
PURGALL,AD=860620.
                 ^-- purge all (your) files not
                     accessed in 2 or more years
                     (assuming today is June 20,
                     1988)
```

PURGE Purge one or more direct or indirect permanent files.

Syntax:

```
PURGE,pfn_1,pfn_2,...,pfn_n/UN=un,PW=pw,NA.
```

Remarks: If the file is attached, it remains as a local file until RETURNed or LOGOUT.

Similar commands:

Cray:	DELETE
NOS/BE:	BEGIN,PAC; BEGIN,PAHC; BEGIN,PALC; BEGIN,PHC; BEGIN,PLC (all DTRC); PURGE
VMS:	DELETE; PURGE

Examples:

```
PURGE,myobj/NA.    <-- be sure file is not present
DEFINE,myobj.      <-- before creating a new one
                       (this is the equivalent of
                       MSSTORE,...,NA=1 under
                       NOS/BE)
```

QGET Assign a queued file to your job.

Syntax:

```
QGET,JSN=jsn,DC=q,UJN=ujn,FN=lfn.
QGET,jsn,q,ujn,lfn.
```

Parameters: DC=q - the queue containing the file

q	meaning
PR	print
PU	punch
PL	plot
WT	wait
IN	input

(default: WT)

FN=lfn - the local file name to be given to the file

Similar commands: NOS/BE: BATCH,lfn,LOCAL

Examples:

```
SUBMIT,myjob,TO.     -or-     CSUBMIT,....
ENQUIRE,JSN.          <-- get jsn of job
QGET,jsn.             <-- get the file from the
                          wait queue
```

RECLAIM Selectively backup and reload local and permanent files.

Syntax:

```
RECLAIM,p1,p2,...,pn./dir1,opts1/dir2,opts2/...
```

Parameters:

pi - parameter
diri - directive
opti - option

Remarks: No REQUEST is needed for a magnetic tape.

Dump tapes MUST be labelled.

RECLAIM tapes are compatible with PFDUMP and PFLOAD.

See also: See NOS 2 Reference Set Volume 3: System Commands for a 15-page discussion of the RECLAIM utility.

Similar commands: NOS/BE: DUMPF; PFLOAD; BEGIN,SELDUMP; BEGIN,SELLOAD (last two DTRC)
VMS: BACKUP

RECOVER (IAF) Recover a detached job or interrupted terminal session.

Syntax:

```
RECOVER,JSN=jsn,OP=T

RECOVER,jsn,T

RECOVER
```

Parameters: jsn - job sequence number of the detached job

T - abort recovery if no recoverable files (else start a recovery dialog)

Examples:

```
/RECOVER,ABCD
```

REDO (IAF) Recall a previously entered command to modify and re-execute it without having to retype the entire command.

Syntax:

```
REDO,string/GO
R,string/GO
```

Parameters: string - the first up-to-10 characters or the command to be REDOne (a blank or terminator in string ends the command) (default: the most recent command)

GO - re-execute without modification (OLD:, MOD:, NEW: prompts are suppressed)

Edit chars:
space - leave character unchanged
- delete character any shift line to left
& - replace character with a space
^ - insert characters before the marked character (end the inserted string with a #; ^ RETURN displays the command line as edited so far)
! - delete to the end of the line
other - replaces the original character

Similar commands: VMS, VMS Cray Station: <UP arrow>

Examples:

```
/REDO
OLD:  CATLIST,LO=F,FN=ABCDEFG
MOD:                  hij!
NEW:  CATLIST,LO=F,FN=HIJ
= = = = =
/REDO
OLD:  CATLIST,LO=X,FN=ABCDEFG
MOD:             f    hij!^
NEW:  CATLIST,LO=F,FN=HIJ      <-- changes so far
MOD:              ^P
NEW:  CATLIST,LO=FP,FN=HIJ
```

REDUCE (Loader) Turn the reduce flag on or off.

Syntax:

```
REDUCE.        <-- turn reduce flag on

REDUCE(-)      <-- turn reduce flag off
```

Remarks: When on, the loader determines the field length assigned.

When off, you determine the field length with RFL statements.

See also: RFL

Similar commands: NOS/BE: RFL

Examples:

```
FTN5.
LGO.          <-- program executes in the FL needed
RFL,50000.
REDUCE(-)
LGO.          <-- program executes in 50000 words
REDUCE.       <-- next load executes in what is
                  needed
```

RENAME Change the name of a local file.

Syntax:

```
RENAME,nfn1=ofn1,nfn2=ofn2,...,nfnn=ofnn.
```

Parameters: nfni - the new name

ofni - the existing name

Remarks: Does not change the name in the permanent file directory.

Similar commands: NOS/BE: BATCH,lfno,RENAME,lfnn
VMS: no local file concept

Examples:

```
RENAME,that=this.
                ^-- change local file name THIS to
                    THAT
```

REPLACE Purge an indirect file and replace it with a copy of a local file; save a copy of a local file as a new indirect file.

Syntax:

```
REPLACE,lfn_1=pfn_1,lfn_2=pfn_2,...,lfn_n=pfn_n/
           UN=un,PW=pw,M=m,NA.
```

Remarks: If the file already exists, the catalog type (CT=) and all other information about the file is preserved; if it does not, a new file is created with CT=PRIVATE.

See also: SAVE

Examples:

```
REPLACE,mylfn=myprog
             ^-- replaces indirect file MYPROG
                 with the contents of local file
                 MYLFN
```

REQUEST Request a tape be mounted.

Remarks: Use LABEL.

REQUEST Assign a file to receive checkpoint dumps, or send a message to the operator to assign to the described device.

Syntax: REQUEST,lfn,ckpt.comment

Parameters: ckpt - lfn is to be a checkpoint file

ckpt	meaning
CK	put each dump at end of lfn
CB	put each dump at beginning of lfn

comment - message to the operator about device assignment

Similar commands: NOS/BE: LABEL,...,X=CK.

Examples:

```
REQUEST,lfn,CK.
             ^-- save all checkpoints

REQUEST,lfn,CB.
             ^-- save the last checkpoint

REQUEST,lfn1,CB.
REQUEST,lfn2,CB.
             ^-- save consecutive checkpoints by
                 alternating two checkpoint
                 files.

DEFINE,lfn.
REQUEST,lfn,CK.  -or-  ASSIGN
CKP.
             ^-- make checkpoint file permanent
```

RERUN Allow a job to be rerun if necessary.

Syntax: RERUN.

Remarks: A job is normally rerunable unless it does something which might make a rerun fail, such as creating, modifying or deleting a file, etc.

See also: NORERUN

Similar commands: Cray: RERUN

Examples: RERUN.

RESOURC Specify that more than one tape drive is required.

Syntax: RESOURC,rt1=u1,rt2=u2,...,rtn=un

Parameters:

```
rti    - resource type
            LO - 7-track tape,  200 cpi
            HI - 7-track tape,  556 cpi
            HY - 7-track tape,  800 cpi
            HD - 9-track tape,  800 cpi
            PE - 9-track tape, 1600 cpi
            GE - 9-track tape, 6250 cpi

ui     - maximum number of units this job will
         use concurrently
            0 - clear a resource type that is no
                longer required
```

Remarks: Jobs needing only a single tape drive at a time, even for a multi-reel file, do not need a RESOURC statement.

RESOURC should precede the first tape request. Subsequent RESOURC statements may change any rt=u.

This statement helps prevent deadlock.

Similar commands: NOS/BE: job statement parameter
VMS: ALLOCATE

Examples:

```
RESOURC,GE=2.     <-- two 6250 cpi, 9-track tapes
                      are required at once

JOB123.
USER,....
CHARGE,....
RESOURC,PE=2.   <-- 2 1600-cpi tapes needed
LABEL,T1,D=PE,VSN=tape1.
LABEL,T2,D=PE,VSN=tape2.
```

```
   ...
RETURN,T1,T2.
RESOURC,PE=1,GE=1.     <-- 1 1600 and 1 6250
   ...
```

RESTART Restart a checkpointed job.

Syntax: `RESTART,lfn,nnnn,x_i.`

Parameters:

```
lfn  - the checkpoint file
       (must have write permission)

nnnn - number of the checkpoint
       (Default: 1; use * for last checkpoint)

x_i  - RI - do not restore command file on lfn
       NA - do not abort if a required file is
            not available; if read parity while
            restoring a file in checkpoint nnnn,
            use checkpoint nnnn-1
       FC - do not restore files ZZZZZC0, C1, C2,
            if already local
```

Similar commands: NOS/BE: RESTART

Examples: `RESTART,ckpfyl,*.`

RETURN Release files (and file space depending on file type) assigned to a job.

Syntax:

```
RETURN,lfn1,lfn2,...,lfnn.     <-- all listed
                                   files

RETURN,*,lfn1,lfn2,...,lfnn.   <-- all but listed
                                   files
```

Parameters: lfni - a file assigned to your job

See also: CLEAR, EVICT, UNLOAD

Similar commands:

```
Cray:    RELEASE
NOS/BE:  CLEAR; RETURN; RETAIN
VMS:     CLOSE it in a program
```

Examples: `RETURN,dtlib,sublib,work1,out.`

REVERT Return from a procedure.

Syntax: `REVERT,opt.com`

Parameters: opt - revert option

opt	meaning
ABORT	return to next EXIT, unless NOEXIT (REVERT appears at your terminal and in the job dayfile)
EX	return to calling procedure and execute command <com> (REVERT appears in the job dayfile but not at your terminal)
NOLIST	return to calling procedure (REVERT does not appear at your terminal or in the job dayfile)

com - for opt EX: the command to be executed
for other opt: a comment

Remarks: The following statements are supplied automatically at the end of a procedure to insure that a REVERT is present:

```
$REVERT.CCL
$EXIT.CCL
$REVERT,ABORT.CCL
```

Similar commands:

Cray:	RETURN
NOS/BE:	REVERT
VMS:	EXIT

Examples:

```
.PROC,MYPROC.
.* the body of your procedure
REVERT,NOLIST.
EXIT.
DMP,30000.
REVERT,ABORT.
```

REWIND Position files at beginning-of-information (BOI).

Syntax:

```
REWIND,lfn1,lfn2,...,lfnn.      <-- all listed
                                    files

REWIND,*,lfn1,lfn2,...,lfnn.    <-- all but listed
                                    files
```

Parameters: lfni - a file assigned to your job

Similar commands: Cray, NOS/BE: REWIND

Examples: `REWIND,myfile.`

RFL Set field length for the next program execution.

Syntax:

```
RFL,CM=nnnnnn.
RFL,nnnnnn.
```

Remarks: nnnnnn may not exceed the last MFL or job statement setting.

See also: MFL, REDUCE

Similar commands: NOS/BE: EFL, RFL

Examples:

```
FTN5.
LGO.
RFL,50000.
REDUCE(-)
LGO.
REDUCE.
```

ROUTE Direct the disposition of an indirect file and define its characteristics.

Syntax:

```
ROUTE,lfn,parameters.
```

Parameters:

```
DC=dc    - disposition code
             IN - input queue
             LP - any printer
             PL - plot
             PR - same as LP
             PU - punch coded
             P8 - punch 80-column binary
             SB - punch system binary
             SC - rescind prior routine and make
                  the file type local
             TO - input queue (output to wait
                  queue)
            (default: same as in previous ROUTE
                      for this lfn; if none, DC=SC,
                      except these special names:
                        name     DC
                       ------    --
                       OUTPUT    PR
                       PUNCH     PU
                       PUNCHB    SB
                       P8        P8  )
DC       - same as DC=SC

DEF      - defer routing until end-of-job
           (default: do it now;
            not allowed with DC=IN, NO, TO)
```

EC=ec - external characteristics for print and punch files

print:

ec	meaning
A6	ASCII graphic 63/64-char set
A9	ASCII graphic 95-char set (lfn must be 7-bit ASCII8)

FC=fc - forms code
FC - use standard print or card forms

FID=ujn - NOS/BE compatibility (same as UJN=)

REP=rep - number of extra copies (default: 0 (only 1 copy printed); maximum: 31 (37B))

TID= - see UN

UJN=ujn - user job name for the file (not input)

UN=un - user name of the receiving remote batch or interactive user
UN - implicit remote routing

Remarks: In general, if a parameter is omitted, it retains the definition from the last ROUTE command which referenced that "lfn". The exception is DEF.

Similar commands:

Cray:	DISPOSE
NOS/BE:	ROUTE
VMS:	SUBMIT; PRINT; XEROX (DTRC); FICHE (DTRC)

Examples:

```
CATLIST,LO=F,L=out1.
ROUTE,out1,DC=PR.        <-- print at Central Site
   -or-
ROUTE,out1,DC=PR,UN=ANAP.
                         ^-- print at Annapolis
= = = = =
ROUTE,mydata,DC=PU,FC=NP,UJN=xxxxx.
                         ^-- punch deck with banner
                             card of "xxxxx"
```

RTIME Put the real-time clock time into the dayfile.

Syntax: RTIME.

See also: CTIME, STIME

```
Similar commands:  Cray:
                   NOS/BE:  DFDATIM; PTIM (both DTRC)
                   VMS:     ^T

Examples:     RTIME.
```

SATISFY (Loader) Satisfy unsatisfied externals now, instead of at the end of the loading.

```
Syntax:       SATISFY.
              SATISFY,lib1,lib2,...,libn.

Parameters:   libi - a specific library to be searched in the
                     listed order
                     (default: all known libraries are searched)

Similar commands:  NOS/BE:  SATISFY

Examples:     LOAD,bin1.
              SATISFY(mylib)
              LOAD,bin2.
              SATISFY.
              bin3.
```

SAVE Put a copy of a local file on disk as an indirect file.

```
Syntax:       SAVE,lfn_1=pfn_1,lfn_2=pfn_2,...,lfn_n=pfn_n
                  /PW=pw,CT=ct,M=m,SS=ss,BR=br,PR=pr,NA,AC=ac.

See also:     REPLACE

Examples:     SAVE,mytemp=keepit/CT=PU.
                             ^-- save local file MYTEMP as a
                                 public file unless KEEPIT
                                 already exists
```

SCOPY Copy coded file(s) displaying EORs and EOFs in the receiving file.

```
Syntax:       SCOPY,lfn_in,lfn_out,n,fchar,lchar,na,R,fcs,
                    fline,lline,ns.

Parameters:   n      - decimal number of files to copy
                       (default: copy to EOI)

              fchar  - first character position of line to copy

              lchar  - last character position of line to copy
```

na - do not abort if no line terminator before EOR

R - rewind lfnin and lfnout before copying

fcs - character set of lfnin
0 - display code or 6/12-bit display code
(default: 0; no other value allowed)

fline - first line of (sequenced) file to be copied
(default: 1)

lline - last line of (sequenced) file to be copied
(default: controlled by n)

ns - any non-null value to suppress EOR/EOF display in lfnout

Remarks: Do not use with S, L, F or SI tapes.

A file without an EOR will have one added to the end of the listing.

See also: COPY (S, L, F tapes), TCOPY (SI tapes)

Similar commands: NOS/BE: LISTEOI; LISTZ (both DTRC)

Examples: SCOPY,myfile.

SCREEN (IAF) Set your terminal for screen mode.

Syntax: SCREEN,TM=model.

SCREEN,model.

Parameters: model - the terminal mode

model	meaning
DT100	DEC VT100-compatible for FSE at DTRC
VT100	alternate DEC VT100-compatible
other	call User Services
omitted	no change

modelT - append T for type-ahead capability

Remarks: SCREEN may be included in a procedure.

Affects FSE, HELPME, screen formatting, and the display of NOS procedure parameters.

You may wish to put "SCREEN,DT100." into your LOGINPR file.

See also: LINE

Examples:

```
SCREEN,DT100      <-- set for full-screen editing
FSE,myfile,G.     <-- edit in full-screen mode
```

SET Assign a value to a control register, an error flag, or the enter-skipped-commands-in-the-dayfile flag; change the current interactive subsystem.

Syntax: SET,symb1=exp1,symb2=exp2,...,symbn=expn.

Parameters: symbi - one of:

name	meaning
R1, R2, R3	local control registers (initial value: 0)
R1G	global control register (initial value: 0)
EF	local error flag (initial value: 0)
EFG	global error flag (initial value: 0)
DSC	dayfile_skipped_command flag 0 - do not put skipped commands into dayfile 1 - put skipped commands into dayfile (initial value: 0)
PL or PS	page length (or page size) (default: 60)
PW	page width (default: 136)
PD	page density (default: 6 lines / inch)
SS	interactive subsystem

expi - any valid expression

symbol	range
R1, R2, R3, R1G	-131071 to 131071
EF, EFG	0 to 63
DSC	1 or 0
PL or PS	16 to 255
PW	40 to 255
PD	6 or 8
SS	ACCESS, BASIC, BATCH, EXECUTE, FORTRAN, FTNTS, NULL

Similar commands: Cray, NOS/BE: SET
VMS: $ var = value (DCL)

Examples:

```
SET,R1=0.
WHILE,R1=5,LOOPEND.
...
SET,R1=R1+1.
ENDW,LOOPEND.
```

SETASL Set the SRU limit for an accounting block.

Syntax:

```
SETASL,s.
SETASL,*.       <-- set to your maximum SRU limit
```

Parameters: s - maximum number of SRUs allowed
(decimal, or octal with B suffix)
(generally, s must be >= the current job step
SRU count and <= your SRU limit)

See also: SETJSL

Examples:

```
SETASL,2000.
```

SETCORE Preset each word of the field length except for RA+2.

Syntax:

```
SETCORE,p.
SETCORE,-p.
```

Parameters: p - desired setting (-p sets the complement of p)

p	fill characters
0	0
ZERO	zeros (0)
INDEF	indefinite (1777 0000 ... 0000)
INF	infinite (3777 0000 ... 0000)

(Default: 0)

Remarks: To preset memory with a load sequence, use LDSET,PRESET.

Examples:

```
RFL,100000.
SETCORE.        <-- immediately clear FL to 0.
```

SETFS Set the auto-drop/no-auto-drop status of files assigned to your job.

Syntax:

```
SETFS,lfn1,lfn2,...,lfnn/FS=fs.
```

Parameters: FS= - auto-drop status

fs	meaning
AD	auto-drop
NAD	no-auto-drop

Remarks: Files with no-auto-drop set are not returned by CLEAR, RETURN(*), or UNLOAD(*).

Examples:

```
SETFS,fyl1,fyl2/FS=NAD.
```

SETJOB Change some of the current job's attributes.

Syntax:

```
SETJOB,UJN=ujn,DC=dc,OP=op.
SETJOB,ujn,dc,op.
```

Parameters: ujn - new job name

dc - disposition code

dc	meaning
TO	queue output with wait disposition
NO	discard output (no dayfile)
DF	default output processing

op - job processing option

op	meaning
SU	job remains suspended until recovered or timed out
TJ	system terminates the job

See also: RECOVER

Examples:

```
SETJOB,xxxx,NO.
```

```
SETJSL   Set the SRU limit for each subsequent job step.

         Syntax:       SETJSL,s.
                       SETJSL,*.       <-- set to your maximum SRU limit

         Parameters:   s - maximum number of SRUs for job step execution

         Examples:     SETJSL,250.


SETPR    Do not use at DTRC.


SETTL    Set the CPU time limit for each subsequent job step.

         Syntax:       SETTL,t.
                       SETTL,*.        <-- set to unlimited

         Parameters:   t - maximum number of CPU seconds for job step
                           execution
                           (default: 64 decimal)

         See also:     ENQUIRE, LIMITS

         Similar commands:  NOS/BE:  ETL (Intercom)

         Examples:     SETTL,5.


SHOW     (IAF) Display a screen formatting panel for testing purposes.

         Syntax:       SHOW,I=panelname.

         Parameters:   I= - the name of a compiled panel in user library
                            PANELIB or in a global library set

         Remarks:      SHOW is an interactive procedure (? for help).
```

SKIP Unconditionally skip succeeding commands, ending with an ENDIF with a matching label.

Syntax: SKIP,label.

Parameters: label - alphanumeric string (1-10 characters, starting with a letter)

See also: ENDIF

Similar commands:

NOS/BE:	SKIP
VMS:	GOTO

Examples:

```
IF(R1<=1,DONE)
   ...
   SKIP(DONE)
IF(R1=2,DONE)
   ...
   SKIP(DONE)
...
ENDIF(DONE)
```

SKIPEI Position a file at end-of-information.

Syntax: SKIPEI,lfn.

Remarks: On magnetic tape with no EOI defined, stops at EOF.

See also: SKIPF, SKIPFB, SKIPR

Similar commands:

Cray:	SKIPD
NOS/BE:	EOI (DTRC)
VMS:	OPEN with ACCESS=APPEND in a program

Examples: SKIPEI,myfile.

SKIPF Skip forward a specified number of files.

Syntax: SKIPF,lfn,n,m.

Parameters: n - decimal number of files to skip (default: 1; max: 262143)

m - coded or binary

m	meaning
B	binary
C	coded

(default: B; C with SI tape is fatal)

```
        Remarks:    Will stop at EOI.

        See also:   SKIPEI, SKIPFB, SKIPR

        Similar commands:  Cray:    SKIPD; SKIPF; SKIPR; SKIPU
                           NOS/BE:  SKIPF

        Examples:   SKIPF,myfile,4,C.
                              ^-- skip 4 coded files


SKIPFB  Skip backward a specified number of files.

        Syntax:     SKIPFB,lfn,n,m.

        Parameters: n - decimal number of files to skip
                        (default: 1; max: 262143)

                    m - coded or binary
                         m   meaning
                         -   -------
                         B   binary
                         C   coded
                        (default: B; C with SI tape is fatal)

        Remarks:    Will stop at BOI.

        See also:   SKIPEI, SKIPF, SKIPR

        Similar commands:  Cray:    SKIPD; SKIPF; SKIPR; SKIPU
                           NOS/BE:  BKSP; SKIPB

        Examples:   SKIPFB,myfile,4,C.
                              ^-- skip back 4 coded files
```

SKIPR Skip forward a specified number of record or file marks.

Syntax: `SKIPR,lfn,n,level,m.`

Parameters:

n - decimal number of files to skip
(default: 1; max: 262143)

level - level number (0-17)
0-16 - EORs and EOFs counted
17 - EOFs counted
(default: 0)

m - coded or binary

m	meaning
B	binary
C	coded

(default: B; C with SI tape is fatal)

Remarks: Consecutive EORs or EOFs are counted separately.

Will stop at EOI.

See also: SKIPEI, SKIPF, SKIPFB

Similar commands:

Cray:	SKIPD; SKIPF; SKIPR; SKIPU
NOS/BE:	SKIPR

Examples:

```
SKIPR,myfile,4,17.
               ^-- skip 4 binary files
```

SLOAD (Loader) Selective load modules from a file.

Syntax: `SLOAD,lfn,name1,name2,...,namen.`

Parameters:

lfn - the file from which the listed modules are to be loaded

namei - the name of a module to be loaded

See also: LIBLOAD

Similar commands:

NOS/BE:	SLOAD
VMS:	LINK file/INCLUDE=

Examples:

```
SLOAD,mybin,suba,subb,subg.
```

SORT This deals with sequenced files and is NOT the Sort/Merge program.

Remarks: SORT5 is the Sort program; MERGE is the Merge program.

SORT5 Sort files.

```
Syntax:       FILE,lfninl,....
              FILE,lfnin2,....
                 ...
              FILE,lfnout,....
              SORT5.pl,p2,...,pn
                 or
              SORT5.pl p2 ... pn

              Positional:      SORT5.from,to,key,dir,l,,e,el,
                                     dialog,end,,,ownf,ownfl,
                                     ownmrl,,own1,own2,own3,
                                     own4,own5,retain,seqa,
                                     seqn,seqr,seqs,status,sum,,
                                     verify,fastio.

              Interactive:     SORT5.DIALOG=YES

              Directive file:  SORT5.DIR=lfn
                               SORT5.params,DIR=lfn
                               SORT5.DIR=lfn,params
                               SORT5.params,DIR=lfn,..
                                    .more_params

Parameters:   FROM=lfn
              FROM=(lfn1,lfn2,...,lfnn)
                    Up to 100 input files, read in the order
                    specified, normally rewound before and
                    after use.

              TO=lfn
                    The file to receive the sorted records,
                    normally rewound before and after use.

              KEY=(key_def,key_def,...)
                    key_def - range                    -or-
                              (range,type,ad)          -or-
                              (first,length,type,ad)
                    range   - first          -or-
                              first..last
                    first   - first byte/bit of key field
                    last    - last byte/bit of key field
```

```
length  - number of bytes/bits in key
          (default: 1)
type    - name of numeric data format
          or collating sequence
          (default: ASCII6)
ad      - order: A (ascending)
                 D (descending)
          (default: A)
Up to 100 key-defs may be specified.
```

Keys are sorted first by the leftmost key_def.

If no keys are specified, KEY=1..mnr (minimum record length; smallest MNR or smallest FL or MRL on FILE statements) is used.

DIR=lfn
DIR=(lfn1,lfn2,...)

Read SORT5 parameters from one or more files.
(default: no directive file is read; the parameters of the SORT5 statement completely define the sort)

L=lfn

Output listing information.
(default: OUTPUT)

E=lfn

Error listing file.
(default: the L= file)

EL=el

```
Error level to be reported:
 T - trivial + W, F, C
 W - warning + F, C
 F - fatal   + C
 C - catastrophic
(default: W)
```

DIALOG=YES or DIA=Y

Interactive dialog. May appear only in the SORT5 control statement. All information for the sort is entered in response to questions. All other parameters specified in the SORT5 statement, except STATUS, are ignored.

ENR=expr
ENR=expr..expr

The estimated number of records to be sorted (single decimal integer 0-10**9, a range of values, or one of the CCL variables: R1, R2, R3, R1G, EF, or EFG. (Use especially if ENR < 1500)

RETAIN=retain or RET=r

Specify the order for records with equal sort keys.

retain	meaning
YES or Y	records with equal keys retain their original order
NO or N	records with equal keys might not retain their original order

(default: NO)

STATUS=variable or ST=variable

Report the SORT5 status to one of the CCL variables: R1, R2, R3, R1G, EF, or EFG.

code	meaning
0	no errors
20	trivial
30	warning
40	fatal
50	catastrophic

Remarks: Each line of the SORT5 control statement or the directive file may be up to 100 characters, but characters beyond column 80 are ignored.

Batch: To continue on more than one line, end one line with two periods and start the next line with one period. CAUTION: because a line range is indicated by two periods, ranges must not be continued.

Interactive: Lines cannot be continued. If more than one line is needed, use a directive file or a procedure.

FILE statements are required for each file. The maximum record is specified with the FL parameter (if RT=Z or F) or MRL parameter (all others).

In the positional illustration above, reserved positions are indicated by adjacent commas.

See Sort/Merge Version 5 Reference Manual, 60484800, for other parameters.

See also: MERGE

```
Similar commands:  Cray, VMS:  SORT
                   NOS/BE:     SORT5
```

```
Examples:    FILE,infyl,BT=C,RT=Z,MRL=80.
             FILE,outfyl,BT=C,RT=Z,MRL=640.
             SORT5.infyl,outfyl,5..10
                         ^-- sort columns 5-10 into ascending
                             ASCII6 order
             SORT5.FROM=infyl,KEY=((5..10,,D)),TO=outfyl
                         ^-- same, except descending
             =====
             SORT5.KEY=6..25
                         ^-- sort one 20-byte key starting in
                             byte 6
             SORT5.KEY=((6,20))
                         ^-- same as above
             SORT5.KEY=(6,20)
                         ^-- sort two 1-byte keys (major key
                             in byte 6, minor key in byte 20)
             SORT5.KEY=6,20
                         ^-- sort byte 6, read directives
                             from local file "20" (this is
                             the next positional parameter)
```

STIME Put the accumulated SRU value for the job into the dayfile.

Syntax: STIME.

See also: CTIME, ENQUIRE,S, RTIME

```
Similar commands:  NOS/BE:  ASSETS, PTIM (DTRC), SUMMARY
                   VMS:     ^T
```

Examples: STIME.

SUBMIT Put a job into the input queue.

Syntax: SUBMIT,lfn,q,NR.c

Parameters: lfn - the file to be submitted - the first record must be in 6-bit display code

```
                q   - output disposition
                       BC or B - central site
                       NO or N - discard output unless
                                 specifically routed -
                                 no dayfile
                       RB=un   - route output to a remote
                                 batch terminal or interactive
                                 user
                       TO      - queued with wait disposition

                NR  - do not rewind the submit file or cREAD file
                      before or after processing
                      (Default: rewind)

                c   - prefix character for reformatting
                      directives in the file (assumes /JOB is the
                      first statement)
                      (Default: /)

        Remarks:     For both direct and indirect files.

        See also:    CSUBMIT, ROUTE

        Similar commands:  Cray:     SUBMIT
                           NOS/BE:   ROUTE,...,DC=IN
                           VMS:      SUBMIT; CRAY SUBMIT

        Examples:    SUBMIT,myjob,BC.
                                  ^-- print at Central Site
                     SUBMIT,myjob,RB=xxxx.
                                  ^-- use QGET to retrieve output from
                                      print queue
                     SUBMIT,myjob,TO.
                                  ^-- use QGET to retrieve output from
                                      wait queue


SWITCH  Set sense switches.

        Syntax:      SWITCH,switch_1,switch_2,...,switch_n,jsn.

        Parameters:  switch_i - switch to be set (1-6)
                                0 - set all switches

                     jsn      - may appear in any parameter position
                                (Default: the current job)

        Similar commands:  Cray, NOS/BE:  SWITCH

        Examples:    SWITCH,1,3,5.
                                  ^-- turn on sense switches 1, 3, 5
```

TCOPY Copy X (binary), E, B, or SI files to disk, I, or SI (binary) tape.

Syntax:

```
TCOPY,lfn_in,lfn_out,format,tc,copycnt,charcnt,
      erlimit,plp2,lfnlst,ns.

TCOPY,I=lfn_in,O=lfn_out,F=format,TC=tc,
      N=copycnt,CC=charcnt,EL=erlimit,PO=plp2,
      L=lfnlst,NS=ns.
```

Parameters:

I= - input file to be copied
(default: INPUT)

O= - the output copied file
(default: OUTPUT)

CC= - maximum number of characters in E or B tape block
(default: 136 (E), 150 (B))

EL= - number of non-fatal errors before abort;
EL=U for unlimited error processing
(default: 0; ignored for E/B output or terminal input)

F= - type of conversion

format	meaning
E	E tape to/from disk, I, SI binary tape (E tape unlabelled and assigned as S)
B	B tape to/from disk, I, SI binary tape (B tape unlabelled and assigned as S)
X	X tape to disk, I, SI binary tape (X tape unlabelled and assigned as S with noise size of 8 frames (7-track) or 6 (9-track))
SI	SI tape to/from disk, I, SI binary tape (SI tape is labelled or unlabelled and assigned as S)

(default: X)

L= - alternate file for parity error messages for EL<>0; cannot be same as I=, O=
(default: OUTPUT)

N= - copy count (meaning determined by TC=)
(default: N=1)

NS= - noise size for E to B conversion
(maximum: 41; NS=0 uses default of 18)

PO= - processing options:

pi	meaning
E	copy input blocks with parity or block-too-long errors (default: error blocks skipped)
T	truncate long blocks for E/B output (default: split into multiple blocks)

TC= - termination condition with N=

tc	meaning
F or EOF	N is number of files
I or EOI	N is ignored (copy to end-of-information)
D or EOD	N is number of double EOFs to copy to

(default: TC=D)

See also: COPY (S, L, F tapes), SCOPY (display EOR/EOF)

Similar commands: Cray:
NOS/BE: COPY; COPYBF; COPYBR; COPYCF;
COPYCR; COPYE; COPYF; COPYR;
COPYRM (last 4 DTRC)

Examples:

```
TCOPY,tape,disk,SI,I.
                 ^-- copy a complete NOS/BE tape
```

TDU (IAF) Compile a terminal definition file and store it in a user library which can later be accessed by a SCREEN or LINE command.

Syntax: TDU,I=definition,L=listing,LIB=library.

Parameters: I= - the terminal definition file in 6/12-bit display code

L= - the output listing file
(default: OUTPUT)

LIB= - the library to receive the load capsule
(default: TERMLIB)

Remarks: TDU may appear in a procedure.

```
TDUMP   Octal or alphanumeric dump of all or part of a file.

        Syntax:      TDUMP,p1,p2,...,pn.

        Parameters:  I=lfn     - local file to be dumped
                                 (default: TAPE1)

                     L=lfn     - listable output (never rewound)
                                 (default: OUTPUT)

                     O         - octal dump only
                     A         - alphanumeric dump only
                                 (if both specified, last is used)
                                 (default: octal and alphanumeric)

                     R=rcount  - decimal maximum number of records to
                                 dump (restarts for each file)
                                 (default: omitted or R=0: dump to EOI)

                     F=fcount  - decimal maximum number of files to dump
                                 (default: dump to EOI; F=0 => dump
                                           until double EOF or EOI)

                     N=ncount  - decimal maximum number of lines to dump
                                 (default: omitted or N=0: dump to EOI)

                     NR        - do not rewind input file

        Similar commands:  NOS/BE:  TDUMP

        Examples:    LABEL,TAPE1,....
                     TDUMP.

TRMDEF  (IAF) Change terminal characteristics.  Use in your prologue
        to set terminal characteristics if you normally use a terminal
        other than the default kind.

        Syntax:      TRMDEF,L=lfn,tc1=v1,...,tcn=vn.

        Parameters:  L=lfn  - listable output

                     tci=vi - new terminal characteristic(s)
                               vi              meaning
                              ----  ------------------------------
                              v     any alphanumeric character
                                    (e.g., $*$; display code 0-44B)
                              $v$   any character (dollar-delimited)
                                    (for "$", user $$$$)
                              vvvB  octal value ASCII character
                                    (e.g., 52B (same as $*$))
```

vvD decimal value of ASCII character (e.g., 42D (same as $*$))

Hvv hexadecimal value of ASCII character (e.g., X2A (same as $*$))

Similar commands: VMS: SET TERMINAL

Examples:

```
TRMDEF,EP=Y. <-- put into LOGINPR to set terminal
                 to half duplex (echo on) for
                 this login
```

ULIB Create a user library; add, delete or replace a record.

Syntax:

```
ULIB,OP=operation,REC=record,LIB=library.

ULIB,operation,record,library.

ULIB?                <-- help about command and
                         prompting for parameters

ULIB,parameter?      <-- help on the parameter and
                         prompting for parameters
```

Parameters: OP= - one of the following operations

operation	meaning
C	create a new user library
A	add a record (same as R)
D	delete a record
R	replace a record (same as A)
F	extract a record and make it local

REC= - name of the record to be added, deleted, replaced, or extracted (must be the name of a local file)

LIB= - the local library file to be created or modified

Remarks: Affects only the local copy. Use SAVE or REPLACE to make the library permanent.

For OP=A, D or R, ULIB returns the original library an creates a new local file -- so ULIB cannot modify a library on a direct file.

Similar commands: Cray, NOS/BE: UPDATE
VMS: CMS; LIBRARIAN;
INCLUDE (in FORTRAN)

Examples: ULIB,C,,mylib.

UNLOAD Release files assigned to your job and perhaps their file space.

Syntax: UNLOAD,lfn1,lfn2,...,lfnn. <-- specified files

UNLOAD,*,lfn1,lfn2,...,lfnn. <-- all but specified files

Similar commands: NOS/BE: UNLOAD

Examples: UNLOAD,myfile.

UNLOCK Rescind the LOCK command and clear the write interlock for specified local disk files.

Syntax: UNLOCK,lfn1,lfn2,...,lfnn.

Remarks: Library files cannot be unlocked.

See also: LOCK

Examples: See LOCK.

UPDATE Create, edit or copy a program library.

Syntax: UPDATE,p1,p2,...,pn.

Parameters: Note: file parameters (C, G, I, K, N, O, S, T) may be followed by 6 (6-bit display code) or 8 (7-bit ASCII -- also requires I8)

pi	meaning
A	copy old sequential PL to new random access PL
B	copy old random access PL to new sequential PL

C=lfn	write compile file on lfn
C=PUNCH	implies D and 8 parameters
C	same as C=COMPILE
omitted	same as C=COMPILE
C=0	no compile output (Note: C is ignored if K is used)
D	compile file is 80 characters
omitted	compile file is 72 characters
E	edit the old PL (to completely edit, use E on two UPDATE commands -- the first will rearrange the directory and remove purged identifiers -- the second will remove identifiers appearing only in the file's directory
F	full Update mode
G=lfn	output file for PULLMOD directives
omitted	appended to the S file
H=n	one of: 3 - use 63-character set 4 - use 64-character set omitted - use old PL char set
I=lfn	primary input directive file
I=0	no input directives
omitted	same as I=INPUT (Note: 6/12_bit ASCII if INPUT is connected, else 6-bit display code)
K=lfn	compile file with decks in order of COMPILE directives
K	same as K=COMPILE
omitted	no input *COMPILE directives
L=c1c2...cn	list options -- one or more of: A - list deck names, correction set identifiers, COMDECK directives, definitions, compile file decks F - same as L=A123456789 (not 0) 0 - no listing 1 - error lines 2 - active UPDATE directives 3 - notes on each line with changed status 4 - text lines

```
                 5 - active compile file
                     directives
                 6 - active and inactive lines
                 7 - all active lines
                 8 - all inactive lines
                 9 - correction history of options
                     5, 7, 8
               Defaults: L=A1234 (correction run)
                         L=A1    (copy)

M=lfn          merge with old PL
M              same as M=MERGE
omitted        no merging
               (Note: both libraries must have the
                      same character set)

N=lfn          new program library
N              same as N=NEWPL
N=0            no new program library
omitted        same as N=0
               (Note: default character set is
                      that of OLDPL (except if
                      OLDPL is 6-bit display code
                      and I file is 7-bit ASCII,
                      then NEWPL is 7-bit ASCII))

O=lfn          listable output file
O=0            no listable output
omitted        same as O=OUTPUT

P=lfn/s1/s2/.../s7
               the old program library
P              same as P=OLDPL
P=0            no old program library
omitted        same as P=OLDPL
               (si are secondary old PLs)

Q              quick mode (process only decks on
               COMPILE directives)

R=c1c2..c4     files to rewind before and after
                 C - compile file
                 N - new PL
                 P - old and merge PLs
                 S - source and PULLMOD files
R              no rewind
omitted        same as R=CNPS (not merge PL)
```

S=lfn	output source file
S	same as S=SOURCE
S=0	no output source file unless T=lfn is specified
omitted	same as S=0
T=lfn	same as S=lfn, except that common decks are excluded (Note: takes precedence over S)
U	do not abort for fatal errors
W	the new PL is sequential
omitted	the new PL is random (except sequential on magnetic tape)
X	compile file is compressed
omitted	compile file is not compressed
8	compile file is 80-character lines
omitted	compile file is 90-character lines
*=char	master control character (any 6-bit octal value 01-50, 53-54)
omitted	master control character is *
/=char	comment control character (A-Z, 0-9, +-*/$=)
omitted	master control character is / (Note: do not use a command abbreviation for <char> unless NOABBREV is in effect)

See also: Section 5-4

Similar commands: Cray, NOS/BE: UPDATE
VMS: CMS; LIBRARIAN

Examples:

```
NOTE,uin./*COMPILE A,B,C
UPDATE,I=uin.
FTN5,I.
RETURN,uin.
EXIT.
RETURN,uin.
= = = = =
/UPDATE.               <-- interactive
? *c a,b,c             <-- enter UPDATE directive(s)
? <CR>                 <-- end-of-file
```

UPROC Specify a user prologue to be executed each time you start a batch or interactive job.

Syntax:

```
UPROC,FN=pfile.
UPROC,pfile.
```

Parameters: pfile - a permanent file with the prologue.
0 - no longer execute a user prologue
(Default: 0)

Remarks: If your prologue is long, you may wish to include RECOVER processing.

LOGINPR is the preferred name.

UPROC is normally executed once to let NOS know that you have such a file.

If you purge your prologue file, future batch jobs will abort.

Similar commands:

NOS/BE:	existence of your file LOGFILEtid
VMS:	existence of your file LOGIN.COM

Examples:

```
/FSE,loginpr.
   < create your procedure >
?? QR      <-- save the file
/UPROC,loginpr.
```

USER Identify you and provide validation information for each batch job.

Syntax:

```
USER,username,password
```

Parameters:

username	- your 4-character User Initials
password	- your 4- to 7-character password

Remarks: This must immediately follow your job statement.

Similar commands:

Cray:	ACCOUNT
NOS/BE:	CHARGE

Examples:

```
USER,xxxx,mypw.
```

VERIFY Compare files in binary mode.

Syntax: VERIFY,lfn1,lfn2,p1,p2,...,pn.

Parameters:

lfn1 - first file to be compared
(default: TAPE1)

lf12 - second file to be compared
(default: TAPE2)

A - abort after completion if errors

BS=bs - maximum block size for S, L tapes
(defaults: 1000B words (S), 2000B (L))

C - both files coded (S, L tapes only)
C1 - first file coded (S, L tapes only)
C2 - second file coded (S, L tapes only)

E=y - maximum number of errors to list
(default: 100)
E - same as E=0

L=lfn - error output file
(default: OUTPUT)

N=x - number of files of multi-file file
(default: 1)
N - compare until EOI on both files
N=0 - compare until empty file in either file

R - rewind both files before and after

Remarks: On mismatch, the record number, word within the record, and the words from both files which do not match are listed.

Your terminal should be in NORMAL mode (not ASCII) before listing VERIFY output at your terminal.

See also: VFYLIB

Similar commands: Cray, NOS/BE: COMPARE

Examples: VERIFY,fyl1,fyl2,N,R.

VFYLIB List differences in name, type, length, and checksum for the records of two library files.

Syntax: VFYLIB,lfn_1,lfn_2,lfn_3,NR.

Parameters: lfn_1 - first file

lfn_2 - second file

lfn_3 - the listable output file

NR - do not rewind lfn_1 and lfn_2 after processing

Defaults: VFYLIB,OLD,NEW,OUTPUT.

Remarks: lfn_1 and lfn_2 are rewound before comparing.

Your terminal should be in NORMAL mode (not ASCII) before listing VFYLIB output at your terminal.

See also: VERIFY (binary comparison)

Similar commands:

Cray:	ITEMIZE
NOS/BE:	ITEMIZE; COMPAR
VMS:	DIFFERENCES

Examples: VFYLIB,fyl1,fyl2,out.

VSN Associate a local file name with one or more volume serial numbers.

Syntax: VSN,lfn_1=vsn_1,lfn_2=vsn_2,...,lfn_n=vsn_n.

Parameters: lfn_i - local file name

vsn_i - 1- to 6-character vsn's to be associated with lfn_i
(\$-delimited if any non-alphanumerics)

vsn_i	meaning
omitted	any available scratch tape is assigned automatically
0	same as omitted
SCRATCH	same as omitted
vsna=vsnb=...=vsnn	duplicate volumes (any may be used)
vsna/vsnb/.../vsnn	successive volumes (must be used in listed order)

See also: LABEL

Similar commands: NOS/BE: VSN

Examples:

```
VSN,tape=NA9876.
```

WHATJSN (IAF) Get the job sequence number for the specified user name.

Syntax:

```
WHATJSN,username
```

Parameters: username - the username whose jsn is desired

Remarks: You must be in the ACCESS subsystem.

Similar commands: NOS/BE: SITUATE
VMS: SHOW USERS

Examples:

```
ACCESS
WHATJSN
DIAL,jsn,message
= = = = =
WHATJSN,xxxx  <-- xxxx is someone's user initials
```

WHILE Start of a command loop.

Syntax:

```
WHILE,exp,label.
```

Parameters:

```
exp   - any valid expression evaluating to true
        or false

label - alphanumeric string (1-10 characters,
        starting with a letter)
```

Remarks: The loop ends with an ENDW statement.

See also: ENDW

Similar commands: Cray: LOOP
NOS/BE: WHILE

Examples:

```
WHILE,R1<5,DOIT.
   ...
SET,R1=R1+1.
ENDW,DOIT.
```

WHO (IAF, DTRC) List the users currently logged in.

Syntax: WHO,lfn.

Parameters: lfn - output file
(default: OUTPUT)

Remarks: The display shows the total number of users who have logged in since the last system deadstart, the number of active (logged in) users (the "D" suffix indicates a decimal number), and a table showing the IAF connection number, User Initials, Job Sequence Number, and port number for each logged-in user. An asterisk in the W column indicates the user has been sent a Warning message by the operator but has not yet received it (messages from the operator while you are in FSE are not received until you exit from FSE).

Similar commands: NOS/BE: SITUATE
VMS: SHOW USERS

Examples:

```
WHO,whoout.   <-- write the display in local file
                  WHOOUT
- - - - -
WHO           <-- display at the terminal
TOTAL USERS =     36D    ACTIVE USERS =     5D

CONN      USER       JSN   W     TERM

   3    CARA       AADD         T1200
   4    TLIB       AAFH         T1210
   5    CASG       AAFW         T1230
   6    CTSC       AAFX         T1240
   7    AMDS       AAFZ         T1610
WHO COMPLETE.
```

WRITEF Write a specified number of file marks on a file.

Syntax: WRITEF,lfn,x.

Parameters: lfn - the file to receive the file marks

x - decimal number of file marks to write
(default: 1; max: 262143)

Remarks: If previous write was not an EOR, one is added.

See also: WRITER

```
         Similar commands:  NOS/BE:  COPYBF;COPYCF;COPYF an empty file

         Examples:    WRITEF,myfile,2.


WRITER   Write a specified number of empty records on a file.

         Syntax:      WRITER,lfn,x.

         Parameters:  lfn - the file to receive the empty records

                      x   - decimal number of empty records to write
                            (default: 1; max: 262143)

         See also:    WRITEF

         Similar commands:  NOS/BE:  COPYBR,COPYCR,COPYR an empty
                                     record

         Examples:    WRITER,myfile,3.


X        (IAF) Execute a batch command.

         Syntax:      X,ccc

         Parameters:  ccc - any valid batch command (up to 80 chars)

         Examples:    X,BASIC      <-- compile a BASIC program without
                                       changing to the BASIC subsystem


X,BASIC  Compile a BASIC program without changing to the BASIC
         subsystem.

         Syntax:      X,BASIC,I=lfn,B=lfn.

         Parameters:  I=      - the BASIC source program
                      omitted - same as I=INPUT

                      B=      - the output object module
                      omitted - execute without creating object module

         Similar commands:  NOS/BE:  BASIC
                            VMS:     RUN (in the BASIC subsystem)

         Examples:    X,BASIC,I=mybas.
                                   ^-- compile and execute (do not
                                       create an object module)
                      = = = = =
                      X,BASIC,I=mybasi,B=mybaso.    <-- compile
                      mybaso.                       <-- execute
```

XMODEM (IAF) Transfer a file between NOS and a PC using the Christensen protocol.

Syntax:

```
XMODEM,fn,td,ft,lf,sp,ec,fm,cf.

XMODEM,FN=fn,TD=td,FT=ft,LF=lf,SP=sp,EC=ec,
      FM=fm,CF=cf.
```

Parameters: Required (if omitted, you will be prompted for them):

fn - file to be transferred

td - transfer direction

td	meaning
S	send from CYBER to micro
R	send from micro to CYBER

ft - file type

ft	S/R	meaning
T	S/R	text - 6-bit display code
A	S/R	text - 6/12-bit display code
E	S/R	text - 8/12-bit ASCII
B	S/R	CYBER binary
M	S/R	micro binary
S	R	automatic based on special characters in the first block

Similar commands:

NOS/BE:	XMODEM (DTRC)
VMS:	use KERMIT

Examples:

```
HELPME,XMODEM.
```

/ABORT (ICF) Abort an interactive Cray job.

Syntax: /ABORT (/AB)

Remarks: May also use USER-BREAK-2 key.

Similar commands: VMS CRAY context: ABORT, DROP, KILL

Examples: /AB

/ATTENTION (ICF) Send an attention interrupt to the interactive Cray job.

Syntax: /ATTENTION (/AT)

Remarks: May also use USER-BREAK-1 key.

Similar commands: VMS CRAY context: ATTENTION

Examples: /AT

/BYE (ICF) Terminate the interactive Cray session.

Syntax:
```
/BYE HOLD AP=NAM_application     (/B)
/BYE QUIT AP=NAM_application
```

Parameters: same as /LOGOFF

Remarks: Equivalent to LOGOFF.

Similar commands: VMS CRAY context: BYE, QUIT

Examples: /BYE

/CONNECT (ICF) Logically connect to another terminal (such as a plotter).

Syntax: /CONNECT terminal_name (/C)

Parameters: terminal_name

See also: /ENDCONNECT, /ICFSTATUS

Similar commands: VMS CRAY context: ATTACH

/DISCARD (ICF) Discard all output sent to the terminal.

Syntax: /DISCARD (/D)

Similar commands: VMS CRAY context: DISCARD

Examples: /D

/ENDCONNECT (ICF) Terminate the logical connection between a master and slave terminal.

Syntax: /ENDCONNECT (/ENDC)

See also: /CONNECT

Examples: /ENDC

/ENDPLAY (ICF) Terminate reading from a PLAY file.

Syntax: /ENDPLAY (/ENDP)

See also: /PLAY

Examples: /ENDP

/EOF (ICF) Send an end-of-file to COS.

Syntax: /EOF (/EO)

Similar commands: VMS CRAY context: EOF

Examples: /EOF

/HELP (ICF) Display a brief description of an interactive Cray command.

Syntax: /HELP command_name (/H)

Parameters: command_name - command for which help is requested (default: a list of all interactive commands and their parameters)

Similar commands: VMS CRAY context: HELP

Examples: /H EOF

/ICFSTATUS (ICF) Display the status of stations and terminals connected to ICF.

Syntax: /ICFSTATUS (/I)

Similar commands: VMS CRAY context: ISTATUS

Examples: /I

/LOGOFF (ICF) Terminate an interactive Cray session.

Syntax:
```
/LOGOFF HOLD AP=NAM_application     (/LOGOF)
/LOGOFF QUIT AP=NAM_application
```

Parameters: HOLD - suspend the interactive session

QUIT - quit the interactive session (default: QUIT)

AP= - the next application

Remarks: same as /BYE

See also: /LOGON

Similar commands: VMS CRAY context: BYE; QUIT

Examples:
```
/LOGOFF
/LOGOFF  HOLD  AP=ICF
```

/LOGON (ICF) Start an interactive Cray session (or reconnect to an existing session).

Syntax: /LOGON MF=mf

Parameters: MF= - the Cray mainframe

mf	meaning
MCR	Cray X-MP at DTRC

Remarks: LOGON is not allowed if you are already logged on.

At the exclamation prompt (!), enter an ACCOUNT statement.

See also: /LOGOFF

Similar commands: VMS CRAY context: INTERACTIVE
VMS: CINT; CRAY INTERACTIVE

Examples: /LOGON

/PERIOD (ICF) Control automatic addition of a period terminator to Cray commands.

```
Syntax:      /PERIOD ON      (/PE)
             /PERIOD OFF

Parameters:  ON  - ICF supplies the terminating period on
                   Cray commands
             OFF - you must supply the terminating period
                   (default: OFF)

Examples:    /PE ON
```

/PLAY (ICF) Read commands and data from a NOS file.

```
Syntax:      /PLAY filename NOECHO     (/PL)

Parameters:  filename - a NOS 8/12-bit ASCII file

             NOECHO   - N - do not echo the lines as they are
                            read
                            (default: echo the lines)

Remarks:     /ENDPLAY

Similar commands:  VMS CRAY context:  @, PLAY

Examples:    /PLAY myascii
```

/PREFIX (ICF) Change the ICF prefix character.

```
Syntax:      /PREFIX prefix_character     (/PR)

Parameters:  prefix_character - the new ICF command prefix
                                character

Remarks:     The default prefix character is slash (/).

             To restore to the default, use "pPREFIX=/",
             where "p" is the current prefix character.

Similar commands:  VMS CRAY context:  none

Examples:    /PR=~      <-- change to tilde
             ~ST        <-- display status
             ~PR=/      <-- restore to slash
             /ST        <-- display status
```

/QUIT (ICF) Immediately terminate the interactive Cray session.

```
Syntax:      /QUIT HOLD AP=NAM_application     (/Q)
             /QUIT QUIT AP=NAM_application
```

Parameters: same as /LOGOFF

Remarks: Can also use LOGOFF or BYE.

Similar commands: VMS CRAY context: BYE; QUIT

Examples: /QUIT

/RESUME (ICF) Resume a suspended interactive Cray session.

```
Syntax:      /RESUME     (/R)
```

See also: /SUSPEND

Examples: /R

/STATUS (ICF) Display Cray job status.

```
Syntax:      /STATUS     (/ST)
```

Similar commands: VMS CRAY context: ISTATUS; JSTAT; STATUS

Examples: /ST

/SUSPEND (ICF) Suspend an interactive Cray session.

```
Syntax:      /SUSPEND     (/SU)
```

See also: /RESUME

Examples: /SU

/* (ICF) An ICF comment line.

```
Syntax:      /* comment
```

Parameters: comment - optional text

Similar commands: VMS CRAY context: COMMENT, MESSAGE

Examples: /* This is a comment

***** Appendix E *****

*** Command Comparison ***

The following is a list of the CDC NOS/BE commands and their equivalent or approximation in CDC NOS, Cray COS, and DEC VAX/VMS.

NOS/BE	NOS	COS	VMS	NOS/BE Description
ACCRPT	-	-	-	DTRC Computer Accounting Worksheet
ADPCOST	-	-	ADPCOST	DTRC Computer Accounting Status Information
ALTER	REPLACE	-	-	Shorten or lengthen an attached permanent file
ASSETS	ENQUIRE LIMITS STIME	STATUS	^T SHOW	(Intercom) Display your terminal status
ATTACH	ATTACH GET	ACCESS ACQUIRE	-	Make a previously cataloged file local
AUDIT	CATLIST	AUDIT	DIRECTORY	Obtain an unsorted list of permanent files
BANNER BANNER3 BANNER6	BLOCK	-	VSYS:BANNER VSYS:BANNER3 VSYS:BANNER6	Create a 10- or 6-line-high banner
BASIC	X,BASIC	-	BASIC	Compile a BASIC program
BATCH	QGET RENAME ROUTE SUBMIT	SUBMIT	SUBMIT	File manipulation
BEGIN	BEGIN	CALL procname	@filename	Transfer control to a procedure
BEGIN,COMQ	BEGIN,FICHE	-	FICHE	Send a file to the microfiche
BEGIN,DOCGET	HELP HELPBE HELPME	-	HELP	On-line documentation
BEGIN,GRIPE	BEGIN,GRIPE	-	GRIPE	Gripe or make suggestions directly to the computer
BEGIN,RENAMAC	BEGIN,NEWCHRG	NEWCHRG	-	Rename permanent file account number
BEGIN,SELDUMP	RECLAIM	-	BACKUP	Selectively dump files to magnetic tape
BEGIN,SELLOAD	BELOAD RECLAIM	-	BACKUP	Selectively load files from magnetic tape

NOS/BE	NOS	COS	VMS	NOS/BE Description
BEGIN,XEROX	BEGIN,XEROX	-	XEROX	Send a file to the Xerox
BKSP	SKIPFB BKSP	SKIPR	-	Backspace a file to read the previous logical record
CATALOG	DEFINE REPLACE SAVE	SAVE	-	Enter a file into the Permanent File Directory
CHARGE	CHARGE USER	ACCOUNT	-	Identify the user and job order number for charging the job
CKP	CKP	-	-	Checkpoint a job
CLEAR	CLEAR RETURN,* UNLOAD,*	-	-	Unload all files except INPUT and OUTPUT
COBOL	-	-	-	COBOL 68 no longer available
COBOL5	COBOL5	-	COBOL	Compile COBOL 74 source program
COMBINE	PACK	-	-	Concatenate logical records
COMMENT	COMMENT * NOTE./	*	!	Insert comments into a control stream
COMPAR	VERIFY	COMPARE	DIFFERENCES	Compare two text files
COMPARE	VERIFY VFYLIB	COMPARE	-	Compare two files in binary mode
CONNECT	ASSIGN	-	-	(Intercom) Connect a file to your terminal
COPY	COPY COPYEI TCOPY	COPYD	COPY	Copy a file to EOI or double EOF
COPYBF	COPYBF COPY TCOPY WRITEF	COPYF	COPY	Copy binary files
COPYBFR	FORM	-	-	Recreate a "random" file from a sequential file
COPYBR	COPYBR COPY WRITER	-	-	Copy binary records
COPYCF	COPYCF COPY TCOPY WRITEF	COPYF	COPY	Copy coded files

NOS/BE	NOS	COS	VMS	NOS/BE Description
COPYCR	COPYCR COPY WRITER	-	-	Copy coded records
COPYE	COPYEI TCOPY	COPYD	COPY	Make an exact copy from the current position to EOI
COPYEXT	L072	-	VSYS:CPYEXT	Reformat a text file
COPYF	COPYBF TCOPY WRITEF	COPYF	-	Copy files or write EOFs
COPYL	COPYL	BUILD	LIBRARIAN	Selective replace of object modules
COPYLM	COPYLM	BUILD	LIBRARIAN	Selective replace of object modules
COPYN	-	BUILD	LIBRARIAN	Copy, merge, or select logical records from up to 10 binary files
COPYR	COPYBR COPYCR WRITER	-	-	Copy records or write EORs
COPYRM	COPY FCOPY FORM	-	RFTAPE WFTAPE	Copy and convert records in sequential files from one record type and block structure to another
COPYS	-	COPYD COPYF	-	A general purpose copy utility for sequential or random files
COPYSBF	COPYSBF LIST	-	-	Copy and shift files
COPYSF	COPYSBF	-	-	Copy and shift files
COPYSR	-	-	-	Copy and shift records
DAY DAYFILE	ERRMSG DAYFILE	ECHO	SET VERIFY	Control the display of dayfile messages at your terminal
DISCARD	PURGE	DELETE	DELETE PURGE	(Intercom) Purge and return a file
DISCONT	ASSIGN	-	-	(Intercom) Disconnect a file from your terminal
DISPLAY	DISPLAY	PRINT	WRITE SYS$OUTPUT	Display a register or computed value in the dayfile
DMD	DMD	DUMP	DUMP	Dump memory in octal and character
DMP	DMP	DUMP	DUMP	Dump memory in octal

NOS/BE	NOS	COS	VMS	NOS/BE Description
DROP	DROP PURGE	ABORT /ABORT	STOP	Drop an executing job
DSMOUNT	-	-	-	No private disk packs
DUMPF	RECLAIM	-	BACKUP	Create a backup tape dump or all files on a user device set
EDITLIB	LIBGEN LIBEDIT GTR	BUILD	LIBRARIAN	Create and maintain a library of programs, subprograms, or procedures
EDITOR	FSE	TEDI	EDT TPU (EVE)	(Intercom) Invoke the CDC text editor
EFL	RFL MFL	-	-	(Intercom) Change execution field length
ELSE	ELSE	ELSE ELSEIF	-	Skip within an IF/IFE construct
ENDIF	ENDIF	ENDIF	-	End an IF/IFE construct
ENDW	ENDW	ENDLOOP	-	End a WHILE loop
EOI	SKIPEI	SKIPD	-	Position file at EOI
ERRORS	-	-	-	(Intercom) Display error statements and messages from compilation listing in file OUTPUT
ETL	SETTL	-	-	(Intercom) Change the command time limit
EVICT	DROP PURGE	-	STOP	Remove a job from a queue
EXECUTE	EXECUTE X	-	RUN	(Loader) Complete loading, search libraries for unsatisfied references, generate load map, execute the program
EXIT	EXIT ONEXIT NOEXIT	EXIT	ON condition	(Batch) Control job or CCL procedure execution
EXTEND	APPEND ATTACH	-	-	Add to the end of an attached permanent file
FETCH	ATTACH GET	ACCESS	-	(Intercom) Attach a file
FILE	FILE	ASSIGN	ASSIGN DEFINE	Describe a file's attributes

NOS/BE	NOS	COS	VMS	NOS/BE Description
FILES	ENQUIRE,F	DS	-	(Intercom) Display local, input, executing, output, and punch files
FIND	ENQUIRE,JSN	-	SHOW SYSTEM	(Intercom) Search all queues for specific jobs
FORM	FORM	-	-	File Organizer and Record Manager
FTN4 FTN	FTN4	-	-	Fortran 66 not recommended
FTN5	FTN5	CFT CFT77	FORTRAN	Compile Fortran 77 program
F45	F45	-	-	Convert FTN4 to FTN5
F45IT	-	-	-	Procedure to convert FTN4 tɔ FTN5 without the need to know the F45 parameters
IFE	IF IFE	IF	IF	IF-THEN-ELSE construct
ITEMIZE	ITEMIZE CATALOG	ITEMIZE	LIBRARIAN	List the contents of a binary file
J	ENQUIRE,J	-	SHOW SYSTEM	(Intercom) Search all queues for specific jobs
jobname	ujn RESOURC	JOB	-	(Batch) The first statement of a batch job, reserves resources
KILL	DROP PURGE	ABORT /ABORT	STOP	Kill an executing job
LABEL	LABEL	-	REQUEST MOUNT	Provide label and mounting information about a magnetic tape
LCS	-	-	-	Convert COBOL 4 to COBOL5
LDSET	LDSET	SEGLDR dir	-	(Loader) Set options for the current load
LGO	LGO	$BLD SEGLDR,GO	LINK RUN	(Loader) Load and execute the default compiler binary output file
LIBLOAD	LIBLOAD	-	-	(Loader) Load modules containing specified entry points from a library
LIBRARY	LIBRARY ULIB	SEGLDR dir	LINK	(Loader) Specify a set of global libraries to be searched for externals and programs and the search order
LIMIT	SETJSL	-	-	Control the amount of disk space which may be used at one time during a batch job or interactive session
LISTBIN	CATALOG ITEMIZE	ITEMIZE	LIBRARIAN	List the contents of a binary or procedure file

NOS/BE	NOS	COS	VMS	NOS/BE Description
LISTMF	LISTLB	-	-	List labels on a multi-labelled tape
LISTZ	SCOPY LIST	-	VSYS:LISTN	List a file
LOAD	LOAD	-	LINK	(Loader) List of files whose contents are to be loaded
LOADPF	BELOAD RECLAIM	-	BACKUP	Reload files from system backup tape
LOCK	-	-	-	(Intercom) Control the receipt of oommunication messages at your terminal
LOGIN	LOGIN HELLO	CRAY INTER CINT	-	(Intercom) Log into Intercom
LOGOUT	LOGOUT BYE GOODBYE QUIT	QUIT	LOGOUT	(Intercom) Terminate an interactive session
M	%MS	-	-	(Intercom) Send a message to Central Site operator
MAP	MAP	SEGLDR dir	LINK	(Loader) Specify load map desired
MERGE	MERGE	SORT	SORT	Merge files
MODE	MODE	MODE	-	Mode error bypass is not recommended at DTRC; an attempt to ignore Error Mode 1 may cause Error Mode 0
MOUNT	-	-	-	No private disk packs
MSACCES	-	BEGIN,MSACCES	HFT ACCESS	Establish access to the Mass Storage System
MSAUDIT	CATLIST	-	HFT DIRECTORY	Audit a user's MSS files
MSCHANG	CHANGE	-	HFT CHANGE	Change MSS file attributes
MSFETCH	ATTACH GET	BEGIN,MSFETCH	HFT FETCH	Make a copy of an MSS file as a local file
MSPASSW	PASSWOR	-	HFT PASSWORD	Change your MSS password
MSPERMT	PERMIT	-	HFT PERMIT	Explicitly define or change permissions for a user
MSPURGE	PURGE	BEGIN,MSPURGE	HFT DELETE	Remove an MSS file
MSSTORE	DEFINE SAVE REPLACE	BEGIN,MSSTORE	HFT STORE	Store a local file on the MSS
MYQ	-	-	SHOW QUEUE	(Intercom) Display the contents of the I/O queues

NOS/BE	NOS	COS	VMS	NOS/BE Description
name	name	name	RUN name	(Loader) Load and execute binary program in local file <name>
NETED	FSE	TEDI	EDT TPU (EVE)	Line editor
NOGO	NOGO	-	LINK	(Loader) Complete loading of a program, generate load map, but do not execute
NOTE	NOTE	NOTE	OPEN,WRITE,CLOSE	Create a file with the command line containing the lines for the new file
OVCAP	-	-	-	(Loader) Preceded each overlay capsule
PAGE	LIST	-	-	(Intercom) Examine a local file
PAUSE	-	-	-	Stop job and display a message for the operator; operator must continue or drop the job
PRNTSPY	-	SPY	-	Print out the results of SPY
PRUDMP	TDUMP	DSDUMP	-	Octal and character dump of files
PURGE	PURGE PURGALL	DELETE	DELETE PURGE	Delete a file from the Permanent File Directory
Q	ENQUIRE,JSN DISPLAY,ALL	-	SHOW QUEUE	(Intercom) Search all queues for specific jobs
RATFOR	-	-	-	Rational Fortran not available
REDUCE	REDUCE	-	-	(Loader) Turn the reduce flag on
RENAME	CHANGE	MODIFY	SET PROTECTION RENAME	Change the attributes (name, passwords, cycle, AC) of a permanent file
REQUEST	DEFINE LABEL REQUEST (CKP)	-	-	File residence request
RESTART	RESTART	-	-	Restart a checkpointed job
RESUME	-	-	-	(Intercom) Resume a SECUREd session
RETAIN	RETURN,* CLEAR	-	-	Unload all files except INPUT, OUTPUT, and those specifically listed
RETURN	RETURN	RELEASE	-	Detach a file from a job or interactive session
REVERT	REVERT	RETURN	EXIT	Return from a procedure
REWALL	REWIND,*	-	-	Rewind all files except INPUT, OUTPUT, and those specifically listed

NOS/BE	NOS	COS	VMS	NOS/BE Description
REWIND	REWIND	REWIND	-	Rewind disk or tape files
RFL	RFL MFL	-	-	(Loader) Request a new field length
ROUTE	ROUTE OUT SUBMIT	DISPOSE SUBMIT	PRINT SUBMIT XEROX FICHE	Direct the disposition of a file and define its characteristics
RUN	-	-	FLR	(Intercom) Compile and execute
SATISFY	SATISFY	-	-	(Loader) Satisfy unsatisfied externals now
SCREEN	SET(FSE) TRMDEF	-	SET TERMINAL	(Intercom) Change your terminal's screen size attributes
SECURE	-	-	-	(Intercom) Secure your terminal
SEGLOAD	SEGLOAD	SEGLDR	-	(Loader) Load a segmented program
SEND	DIAL	-	PHONE	(Intercom) Send a message to another user who is currently logged in
SET	SET	SET	$ var = valu	Put a value into a register
SETNAME	-	-	-	No private disk packs
SITUATE	WHO WHATJSN	STATUS	SHOW USERS	(Intercom) Show the interactive and batch terminals currently logged in
SKIP	SKIP	-	-	Skip commands
SKIPB	SKIPFB BKSP	SKIPD SKIPF SKIPU	-	Skip backward in a file
SKIPF	SKIPF SKIPR	SKIPD SKIPF SKIPU	-	Skip forward in a file
SLOAD	SLOAD	SEGLDR dir	-	(Loader) Selectively load modules from a local file
SORT5	SORT5	SORT	SORT	Sort files
SPY	HOTSPOT	SPY	PCA	Gather histograms of program execution for debugging
STORE	DEFINE SAVE REPLACE	SAVE	-	(Intercom) Catalog a file
SUMMARY	ENQUIRE	JOBCOST	SHOW	Put the accounting summary, up to the current point in the job, into the dayfile

NOS/BE	NOS	COS	VMS	NOS/BE Description
SWITCH	SWITCH OFFSW ONSW	SWITCH	-	Set a pseudo-sense switch
SYSBULL	BEGIN,NEWS	-	NEWS	Display a System Bulletin
TAPDMP9	-	-	-	Dump magnetic tape in hexadecimal, octal, or character
TDUMP	TDUMP	-	-	Octal and character dump of 7-track (odd parity) and 9-track SI tapes, or disk files
TRANSF	-	-	-	Subtract 1 from the dependency counter for a job in a dependency set; when the counter reaches 0, the job may begin execution
TRANSPF	-	-	-	No private disk packs
TURNKEY	PASSWOR	ACCOUNT	SET PASSWORD	(Intercom) Change your login turnkey password
UNLOAD	UNLOAD EVICT	-	-	Rewind and unload a tape, detach a file from an interactive session
UPDATE	UPDATE	UPDATE	LIBRARIAN CMS	Create and maintain a library of source programs or data
VSN	VSN	-	-	Identify the magnetic tape reels to be used in a job
WARNING	-	-	-	(Batch) Create a banner page with a job's classification
WHILE	WHILE	LOOP	-	Loop construct
XEQ	LDSET	-	-	(Intercom) Execute a program requiring one or more Loader commands which cannot be entered directly from a terminal
.*	.*	.*	$!	Comment within a procedure
%A	%2	^Z/INTER	^Y	Abort the executing command
%S	%1	-	^O	Suppress the rest of the current output buffer

***** Appendix F *****

*** References ***

The following manuals describe various features of the Cray, DEC and CDC systems.

** Cray **

SR-0009	Fortran (CFT) Reference Manual
SR-0011	COS Version 1 Reference Manual
SR-0013	UPDATE Reference Manual
SR-0018	CFT77 Reference Manual
SV-0020	DEC VAX/VMS Station Reference Manual
SR-0035	CDC NOS Station Reference Manual
SR-0039	COS Message Manual
SR-0060	Pascal Reference Manual
SR-0066	SEGLDR Reference Manual
SR-0113	Programmer's Library Reference Manual

** DEC **

AA-D034D-TE	Programming in VAX Fortran
AA-Z200C-TE	DCL Dictionary
AA-Z300A-TE	EDT Reference Manual
AI-Y517A-TE	VAX/VMS User's Manual

** CDC NOS **

60460420	NOS Full Screen Editor
60459680	NOS 2 Reference Set Volume 3: System Commands

** General **

CMLD-87-07	Fortran 77 Extensions - A Comparison
CMLD-88/14	Computer Center Reference Manual (this manual)
CMLD-88/15	CDC NOS Full Screen Editor (FSE) User's Guide

***** Appendix G *****

*** CCF Computer Systems ***

Cray

C1

Computer:	Cray X-MP/24
Front ends:	DEC VAXcluster (V3), CDC CYBER 180/860A (N1)
Links to:	Mass Storage System (N1)
Operating system:	COS level 1.16
Services:	batch, timesharing
Schedule:	24 hours a day, 7 days a week, except a few hours Tuesday and Thursday mornings for maintenance
Location:	Central site

DEC VAXcluster

DT1 (V1)

Computer: VAX 11/780
Links to: CDC CYBER 180/860A (N1/MFN); DECnet to DTRC/Annapolis (RM1), NAVAIR (HORNET), NAVSEA (SEAHUB, etc.); DDN-TELNET (TOFACS, etc.)
Operating system: VMS 4.6
Services: batch, timesharing
Schedule: 24 hours a day, 7 days a week, except a few hours Thursday morning for maintenance
Location: Central site

DT2 (V2)

Computer: VAX 11/780
Links to: CDC CYBER 180/860A (N1/MFN), DECnet to DTRC/Annapolis (RM1), NAVAIR (HORNET), NAVSEA (SEAHUB, etc.)
Operating system: VMS 4.6
Services: batch, timesharing
Schedule: 24 hours a day, 7 days a week, except a few hours Thursday morning for maintenance
Location: Central site

DT3 (V3)

Computer: VAX 8550
Links to: Cray X-MP (C1); CDC CYBER 180/860A (N1/MFN); DECnet to DTRC/Annapolis (RM1), NAVAIR (HORNET), NAVSEA (SEAHUB, etc.)
Operating system: VMS 4.6
Services: batch, timesharing
Schedule: 24 hours a day, 7 days a week, except a few hours Thursday morning for maintenance
Location: Central site

DT4 (V4)

Computer: VAX 8550
Links to: CDC CYBER 180/860A (N1/MFN); DECnet to DTRC/Annapolis (RM1), NAVAIR (HORNET), NAVSEA (SEAHUB, etc.)
Operating system: VMS 4.6
Services: batch, timesharing
Schedule: 24 hours a day, 7 days a week, except a few hours Thursday morning for maintenance
Location: Central site

Remote Mini-sites

RM1 (R1)

Computer:	VAX 8250
Links to:	DECnet to VAXclustetr (DT1-DT4), NAVAIR (HORNET), NAVSEA (SEAHUB, etc.)
Operating system:	VMS 5.0
Services:	RJE terminal with local batch, timesharing
Schedule:	24 hours a day, 7 days a week, except a few hours for maintenance
Location:	Annapolis

Control Data Corporation

MFN (N1)

Computer:	CDC CYBER 180/860A with Mass Storage System
Cray Station ID:	N1
Links to:	Cray X-MP (C1)
Links from:	Cray X-MP (C1), DEC VAXcluster
Operating system:	NOS version 2.5.3 level 688
Services:	trillion-bit storage, local and remote batch, timesharing
Schedule:	24 hours a day, 7 days a week, except a few hours for maintenance
Location:	Central site

Office Automation System composed of:

TOFACSA

Computer: DEC VAX 11/780
Links to: Mass Storage System
Operating system: Ultrix-32
Services: TOFACS Office Automation (primarily Carderock)
Schedule: 24 hours a day, 7 days a week, except a few hours Thursday morning for maintenance
Location: Central site

DTRC

Computer: DEC VAX 11/780
Links to: Mass Storage System
Operating system: Ultrix-32
Services: TOFACS Office Automation (primarily Carderock)
Schedule: 24 hours a day, 7 days a week, except a few hours Thursday morning for maintenance
Location: Central site

TOFACSC

Computer: DEC VAX 11/780
Links to: Mass Storage System
Operating system: Ultrix-32
Services: TOFACS Office Automation (primarily Annapolis)
Schedule: 24 hours a day, 7 days a week, except a few hours Thursday morning for maintenance
Location: Central site

*** Services and Support ***

Accounting for Computer Services: Code 189.3	(202) 227-1910
Hardware: Code 1895	(202) 227-1400
Information, Computer status (recorded message):	(202) 227-3043
Manuals: Software Branch (User Services)	(202) 227-1907
Tape Librarian: Hardware Branch	(202) 227-1967
Training: Software Branch (User Services)	(202) 227-1907
User Services (See below)	

Software Branch (User Services)

Carderock: Code 1893.1		(202) 227-1907
	Autovon:	8-287-1907

Stan Willner (Head)
Kevin Brady
Sharon Good
Brenda Peters

Annapolis: Code 1893.1		(301) 267-3343
	Autovon:	8-281-3343

Dave Sommer

Administrative Personnel

Head, Computation, Mathematics and Logistics Department: Dr. Charles Schoman, Code 18	(202) 227-1504
Head, Computer Facilities Division: Gil Gray, Code 189	(202) 227-1270
Head, Computer Accounting: Jean Morris, Code 189.3	(202) 227-1361
Head, Software Branch: Lorraine Minor, Code 1893	(202) 227-1428
Head, Hardware Branch: Albert Glover, Acting Code 1895	(202) 227-1346

***** Appendix H *****

*** Internal Data Structure ***

1. The following table summarizes word lengths on various computers:

computer	op sys	bits/word	digits/word	characters/word
Cray X-MP		64	22 octal	8
CDC CYBER 200		64	16 hex	8
CDC CYBER 180	NOS/VE			
CDC CYBER 180	NOS & NOS/BE	60	20 octal	10
CDC CYBER 170	NOS/BE			
DEC VAX		16	4 hex	2
(when used in Fortran)		32	8 hex	4
IBM		32	8 hex	4
Burroughs 7700		48	12 hex	6
Unisys 1100		36	12 octal	4 (ASCII) 6 (Fieldata)

This affects the conversion of programs in four areas:

a. The degree of precision of operations is different. Therefore, convergence factors may need to be increased or decreased in absolute value.

b. Constants and data may need to be changed.

c. Octal and hexadecimal constants used in masking operations are generally affected and require alteration according to their intended use.

d. Since different computers may store a different number of characters per word, DATA statements that store a string of Hollerith characters may position the characters in different relative positions in different words. All variable formats (whether read in as data or created by the programmer) should be checked. Better yet, Fortran programs which store Hollerith data in INTEGER or REAL variables should be changed to use the Fortran 77 CHARACTER variables and never need to worry about this problem again. (You may have to worry about the maximum length of a CHARACTER variable, but not how it is stored.)

2. Internal representation of character data is ASCII in the Cray X-MP and DEC VAX, Display Code in the CDC CYBER, and ASCII, EBCDIC or internal BCD in some other systems.

CHARACTER string	machine	op sys	internal representation
' ' (1 blank)	Cray X-MP		* oct 20 hex
	CDC 170		55
	CDC 180	NOS	55
	DEC VAX		20
'0' (1 zero)	Cray X-MP		* oct 30 hex
	CDC 170		33
	CDC 180	NOS	33
	DEC VAX		30
'FILE48'	Cray X-MP		* oct 46494C463438 hex
	CDC 170		061014053743
	CDC 180	NOS	061014053743
	DEC VAX		3834454C4946
			(8 4 E L I F)

* - the octal representation depends on the position in the word

Hollerith words	machine	op sys	internal machine representation	
<blanks>	Cray X-MP		0200401002004010020040	oct
			2020202020202020	hex
	CDC 170		55555555555555555555	oct
	CDC 180	NOS	55555555555555555555	oct
	DEC VAX		20202020	hex
<zeroes>	Cray X-MP		0300601403006014030060	oct
			3030303030303030	hex
	CDC 170		33333333333333333333	oct
	CDC 180	NOS	33333333333333333333	oct
	DEC VAX		30303030	hex
FILE48	Cray X-MP		0431112304246416020040	oct
			46494C4534382020	hex
	CDC 170		06101405374355555555	oct
	CDC 180	NOS	06101405374355555555	oct
	DEC VAX		454C4946 20203834	hex
			(E L I F 8 4)	<-- 2 words

3. The character sequence for the Cray X-MP and DEC VAXcluster is ASCII. Note that numbers precede letters for alphabetic comparisons. The character sequences for the CDC computers at DTRC are Display Code (NOS: 64-character set; NOS/BE: 63-character set). CDC Fortran uses the Display Code sequence (letters before numbers); CDC COBOL uses the ASCII6 sequence (numbers before letters). DEC VAX uses the ASCII sequence.

4. The CDC CYBER uses some special bit configurations in floating point arithmetic to indicate indefinite and infinite operands. These errors could be caused by referencing program areas not initialized or areas overwritten due to inadequate storage reservation. The CPU will not do any further calculation if it encounters such a number and the job will abort with an error mode 2 or 4.

+ infinity	3777xxxxxxxxxxxxxxxx	oct
- infinity	4000xxxxxxxxxxxxxxxx	
+ indefinite	1777xxxxxxxxxxxxxxxx	
- indefinite	6000xxxxxxxxxxxxxxxx	

where 'x' is any octal digit, usually 0.

5. The word format of integers and floating point numbers differs on the various computers.

Cray X-MP	integer	floating point	
1, 1.0	0000000000000000000001	0400014000000000000000	oct
	0000000000000001	4018000000000000	hex
-1, -1.0	1777777777777777777777	1400014000000000000000	oct
	FFFFFFFFFFFFFFFF	C001800000000000	hex
2, 2.0	0000000000000000000002	0400024000000000000000	oct
	0000000000000002	4002800000000000	hex
4, 4.0	0000000000000000000004	0400044000000000000000	oct
	0000000000000004	4003800000000000	hex
DEC VAX			
1, 1.0	00000001	00004080	hex
-1, -1.0	FFFFFFFF	0000C080	
2, 2.0	00000002	00004100	
4, 4.0	00000004	00004180	
CDC CYBER			
1, 1.0	0000000000 0000000001	1720400000 0000000000	oct
-1, -1.0	7777777777 7777777776	6057377777 7777777777	
2, 2.0	0000000000 0000000002	1721400000 0000000000	
4, 4.0	0000000000 0000000004	1722400000 0000000000	

Note the difference in the format of negative integers (and CYBER floating point) numbers:

Cray X-MP, DEC VAX	CDC CYBER
two's complement of absolute value	one's complement of absolute value

6. Logical variables are represented by:

	Cray X-MP, CDC CYBER	DEC VAX
TRUE	-1	1 in bit 0
FALSE	0	0 in bit 0

7. By default, your program area in central memory is set as follows:

Cray	DEC VAX	CDC NOS	CDC NOS/BE
zero	zero	zero	DEBUG (negative indefinite with addresses and some bits set for CYBER Interactive Debug) See LDSET,PRESET/PRESETA.

*** Internal Representation ***

** Cray X-MP **

Words in the Cray X-MP are 64 bits long. Bits are numbered 0-63 or 63-0.

```
Integer:  bit    0    - the sign bit (0 = positive; 1 = negative) (23)
          bits   1:23 - the absolute value of the integer          (22:0)
          range       - ~ -10**14 to ~10**14

Integer (CFT,INTEGER=64):
          bit    0    - the sign bit (0 = positive; 1 = negative) (63)
          bits   1:63 - the absolute value of the integer          (62:0)
          range       - ~ -10**19 to ~10**19

Real:     bit    0    - the sign of the number                    (63)
          bits   1:15 - the exponent (2000 bias)                  (62:48)
          bits  16:63 - the mantissa                              (47:0)
          range       - ~10**-2466 to ~10**2465
          precision   - ~ 14 decimal digits

Double:   First word:
          bit    0    - the sign of the number                    (63)
          bits   1:15 - the exponent (2000 bias)                  (62:48)
          bits  16:63 - the high order part of the mantissa       (47:0)

          Second word:
          bits   0:15 - unused                                    (63:48)
          bits  16:63 - the low order part of the mantissa        (47:0)
          range       - ~10**-8193 to ~10**8189
          precision - ~ 29 decimal digits
```

** DEC VAX **

Bytes in the DEC VAX are 8 bits long with bits are numbered 7-0. A word (INTEGER*2 in Fortran) is 16 bits long (15-0). A longword (INTEGER or INTEGER*4) is 32 bits long (31-0).

```
Word (INTEGER*2):
          bit   15    - the sign bit (0 = positive; 1 = negative)
          bits  14:0  - the absolute value of the integer
          range       - -32,768 to 32,767

Longword (INTEGER*4):
          bit   31    - the sign bit (0 = positive; 1 = negative)
          bits  30:0  - the absolute value of the integer
          range       - -2,147,483,648 to 2,147,483,647

F_float (REAL*4):
          bit   15    - the sign of the number
          bits  14:7  - the exponent (excess 128)
          bits   6:0 and
                31:16 - the mantissa
          range       - ~.29*10**-8 to ~1.7*10**38
          precision   - ~ 7 decimal digits

D_float (REAL*8, DOUBLE PRECISION):
          bit   15    - the sign of the number
          bits  14:7  - the exponent (excess 128)
          bits   6:0  and
                63:48 and
                47:32 and
                31:16 - the mantissa
          range       - ~.29*10**-8 to ~1.7*10**38
          precision   - ~ 16 decimal digits

G_float (FORTRAN/G_floating):
          bit   15    - the sign of the number
          bits  14:4  - the exponent (excess 1024)
          bits   3:0 and
                63:16 - the mantissa
          range       - ~.56*10**-308 to ~.9*10**308
          precision   - ~ 15 decimal digits

H_float (REAL*16):
          bit   15    - the sign of the number
          bits  14:0  - the exponent (excess 16,384)
          bits 127:16 - the mantissa
          range       - ~.84*10**-4932 to ~.59*10**4932
          precision   - ~ 33 decimal digits
```

** CDC CYBER (NOS, NOS/BE) **

Words in the CDC CYBER 170 and CYBER 180 (when running NOS or NOS/BE) are 60 bits long. Bits are numbered 59-0.

Integer:	bit 59	- the sign bit (0 = positive; 1 = negative)
	bits 58:0	- the absolute value of the integer
Integer:	bit 59	- the sign bit (0 = positive; 1 = negative)
	bits 47:0	- the absolute value of the integer (if used in multiplication or division)
Real:	bit 59	- the sign of the number
	bits 58:48	- the exponent (2000 bias)
	bits 47:0	- the mantissa with the binary point after bit 0

Double: (Double precision is performed in the software, not in the hardware)

First word:

bit 59	- the sign of the number
bits 58:48	- the exponent (2000 bias)
bits 47:0	- the high order part of the mantissa with the binary point after bit 0

Second word:

bit 59	- the sign of the number
bits 58:48	- the exponent (2000 bias)
bits 47:0	- the low order part of the mantissa with the binary point after bit 0

***** Glossary *****

Alphabetic (CDC - NOS and NOS/BE)
The letters A-Z.

Alphabetic (Cray)
$, %, @, and the letters A-Z, a-z.

Alphabetic (DEC)
$, _ (underscore), and the letters A-Z, a-z (upper and lower case are the same).

Alphanumeric
Alphabetic and the digits 0-9.

User initials (userid or username)
The 4-character ID assigned to each user by Code 189.3. This is used to identify jobs, for charge authorization, to identify permanent and MSS files, magnetic tapes, etc.

***** Index *****

Note - Commands, qualifiers and directives are in upper case.

Major references are flagged with an asterisk after the page number, for example, 1-1*.

Initial Distribution

Copies:

12 Director
Defense Technical Information Center (DTIC)
Cameron Station
Alexandria, Virginia 23314

Center Distribution

Copies:

1	18/1809	Shoman, Dr. C. M.
1	1805	Cuthill, E. H.
2	1809S	
1	182	Camara, A. W.
1	184	Schot, J. W.
1	185	Schaffran, R.
1	187	Zubkoff, M. J.
1	189	Gray, G. R.
1	189.2	
1	189.3	Morris, J.
1	1893	Minor, L. R.
1	1893	Strickland, J. D.
150	1893.1	Willner, S. E.
20	1893.1	Sommer, D. V.
1	1895	Glover, A.
1	1896	Annapolis Computer Center
1	522	TIC (C)
1	522.2	TIC (A)
1	93	Patent Counsel